AIML Capstone Project
(Group 7) NLP project - 2

Submitted by

- Amit Kumar
- Geet Gangwar
- Sambhav Maniar
- Vikas Chandra

1. Summary of problem statement, data and findings

In this project we aim to build an NLP based chatbot for Industrial Safety and Health Analytics database.

Understanding of Problem

This data contains industrial data from Brazil's largest industry. By analysing this dataset we're going to identify the severity of the accidents and analyse the cause of these accidents to better understand and create a framework for industrial safety.

This can be integrated with interactive chatbot so that it can help professionals to highlight the safety risk as per the incident description.

Context: The database comes from one of the biggest industries in Brazil and in the world. It is an urgent need for industries/companies around the globe to understand why employees still suffer some injuries/accidents in plants. Sometimes they also die in such an environment.

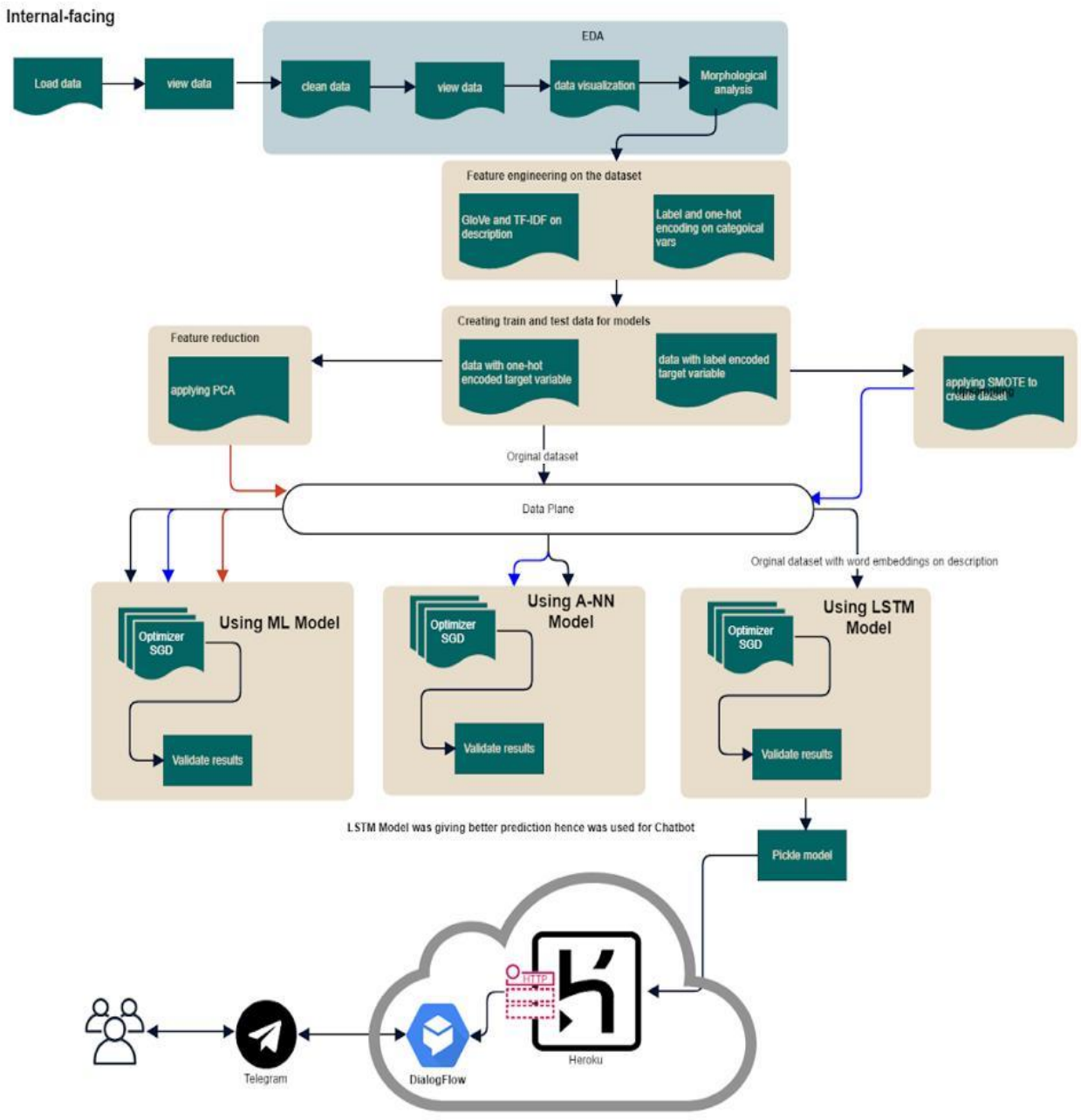
- **DATA DESCRIPTION:**

  This The database is basically records of accidents from 12 different plants in 03 different countries which every line in the data is an occurrence of an accident.

  Columns description:

  - Data: timestamp or time/date information
  - Countries: which country the accident occurred (anonymised)
  - Local: the city where the manufacturing plant is located (anonymised)
  - Industry sector: which sector the plant belongs to
  - Accident level: from I to VI, it registers how severe was the accident (I means not severe but VI means very severe)
  - Potential Accident Level: Depending on the Accident Level, the database also registers how severe the accident could have been (due to other factors involved in the accident)
  - Gender: if the person is male of female
  - Employee or Third Party: if the injured person is an employee or a third party
  - Critical Risk: some description of the risk involved in the accident
  - Description: Detailed description of how the accident happened.

2. Step by Step walkthrough of the solution



As per wiki ”Data preprocessing can refer to manipulation or dropping of data before it is used in order to ensure or enhance performance,[1] and is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects. Data-gathering methods are often loosely controlled, resulting in out-of-range values (e.g., Income: −100), impossible data combinations (e.g., Sex: Male, Pregnant: Yes), and missing values, etc. Analyzing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running any analysis. Often, data preprocessing is the most important phase of a machine learning project, especially in computational biology.”

Approach to EDA and Pre Processing

Data preprocessing is required for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

It involves below steps:

Preview data

	Unnamed: 0	Data	Countries	Local	Industry Sector	Accident Level	Potential Accident Level	Genre	Employee or Third Party	Critical Risk	Description
0	0	2016-01-01 00:00:00	Country_01	Local_01	Mining	I	IV	Male	Third Party	Pressed	While removing the drill rod of the Jumbo 08 f...
1	1	2016-01-02 00:00:00	Country_02	Local_02	Mining	I	IV	Male	Employee	Pressurized Systems	During the activation of a sodium sulphide pum...
2	2	2016-01-06 00:00:00	Country_01	Local_03	Mining	I	III	Male	Third Party (Remote)	Manual Tools	In the sub-station MILPO located at level +170...

- Unamed: 0 columns is dropped

```
df.describe()
```

	Date	Country	Local	Industry Sector	Accident Level	Potential Accident Level	Gender	Employee type	Critical Risk	Description
count	425	425	425	425	425	425	425	425	425	425
unique	287	3	12	3	5	6	2	3	33	411
top	2017-02-08 00:00:00	Country_01	Local_03	Mining	I	IV	Male	Third Party	Others	During the activity of chuteo of ore in hopper...
freq	6	251	90	241	316	143	403	189	232	3

Check total number of entries and column types

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 425 entries, 0 to 424
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Date                      425 non-null    object
 1   Country                   425 non-null    object
 2   Local                     425 non-null    object
 3   Industry Sector           425 non-null    object
 4   Accident Level            425 non-null    object
 5   Potential Accident Level  425 non-null    object
 6   Gender                    425 non-null    object
 7   Employee type             425 non-null    object
 8   Critical Risk             425 non-null    object
 9   Description               425 non-null    object
dtypes: object(10)
memory usage: 33.3+ KB
```

- We noticed that except a 'date' column all other columns are categorical columns.

Check any null values

```
df.isna().sum()
```

```
Date                          0
Country                       0
Local                         0
Industry Sector               0
Accident Level                0
Potential Accident Level      0
Gender                        0
Employee type                 0
Critical Risk                 0
Description                   0
dtype: int64
```

- No null values are present in the dataset.

Check duplicate entries

No. of duplicated entries: 7

	Date	Country	Local	Industry Sector	Accident Level	Potential Accident Level	Gender	Employee type	Critical Risk	Description
76	2016-04-01 00:00:00	Country_01	Local_01	Mining	I	V	Male	Third Party (Remote)	Others	In circumstances that two workers of the Abrat...
77	2016-04-01 00:00:00	Country_01	Local_01	Mining	I	V	Male	Third Party (Remote)	Others	In circumstances that two workers of the Abrat...
261	2016-12-01 00:00:00	Country_01	Local_03	Mining	I	IV	Male	Employee	Others	During the activity of chuteo of ore in hopper...
262	2016-12-01 00:00:00	Country_01	Local_03	Mining	I	IV	Male	Employee	Others	During the activity of chuteo of ore in hopper...
302	2017-01-21 00:00:00	Country_02	Local_02	Mining	I	I	Male	Third Party (Remote)	Others	Employees engaged in the removal of material f...

- There are 7 duplicate entries which will be removed from the dataset.

Check unique values

Unique value in column Country is 3

	Unique value	Count
0	Country_01	248
1	Country_02	129
2	Country_03	41

##

Unique value in column Local is 12

	Unique value	Count
0	Local_03	89
1	Local_05	59
2	Local_01	56
3	Local_04	55
4	Local_06	46
5	Local_10	41
6	Local_08	27
7	Local_02	23
8	Local_07	14
9	Local_12	4
10	Local_11	2
11	Local_09	2

##

Unique value in column Gender is 2

	Unique value	Count
0	Male	396
1	Female	22

##

Unique value in column Industry Sector is 3

	Unique value	Count
0	Mining	237
1	Metals	134
2	Others	47

##

Unique value in column Employee type is 3

	Unique value	Count
0	Third Party	185
1	Employee	178
2	Third Party (Remote)	55

##

Unique value in column Accident Level is 5

	Unique value	Count
0	I	309
1	II	40
2	III	31
3	IV	30
4	V	8

##

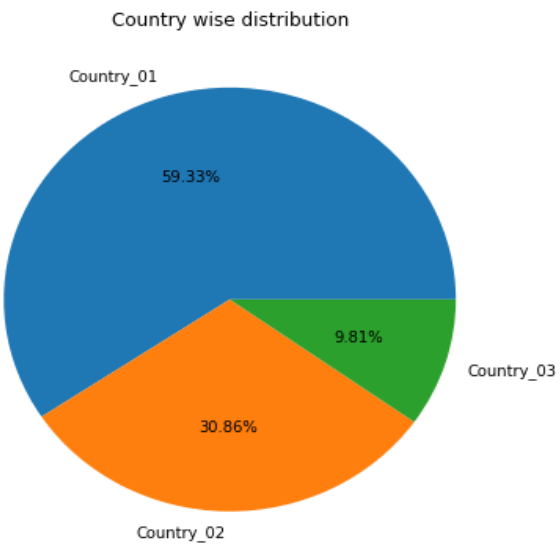
Unique value in column Potential Accident Level is 6

	Unique value	Count
0	IV	141
1	III	106
2	II	95
3	I	45
4	V	30
5	VI	1

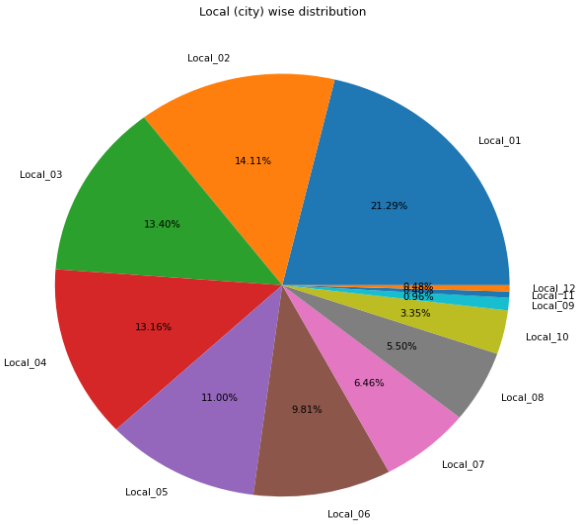
##

Plot count distribution of categorical data (univariate and bi-variate distribution)

Uni-variate Analysis

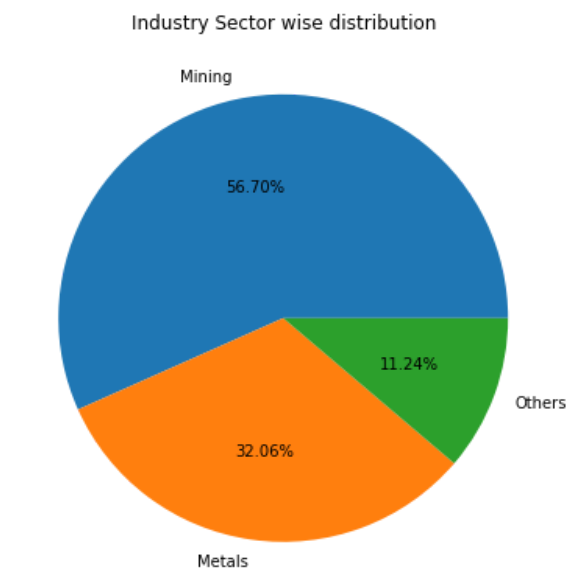


- Country_01 has more accident records.




```
[14] df[df['Local'] == 'Local_03'].Country.value_counts()

     Country_01    90
     Name: Country, dtype: int64
```

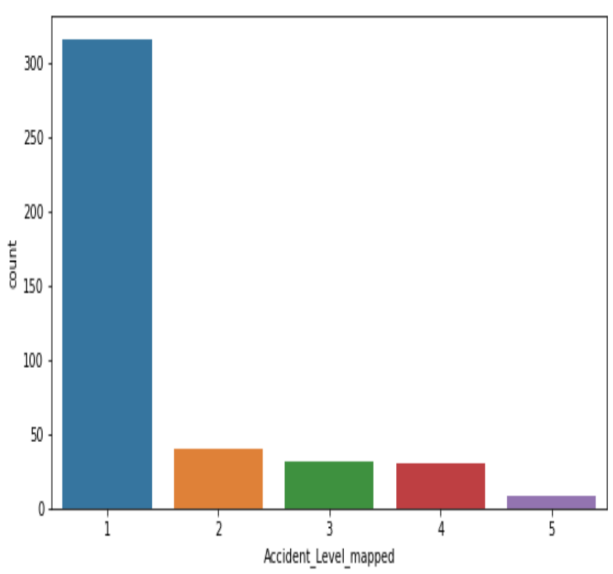
- Most of accidents are reported in Local_03 which belongs to Country_01.



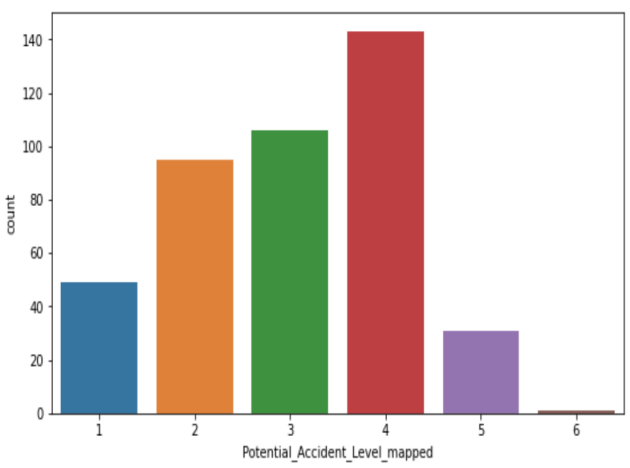
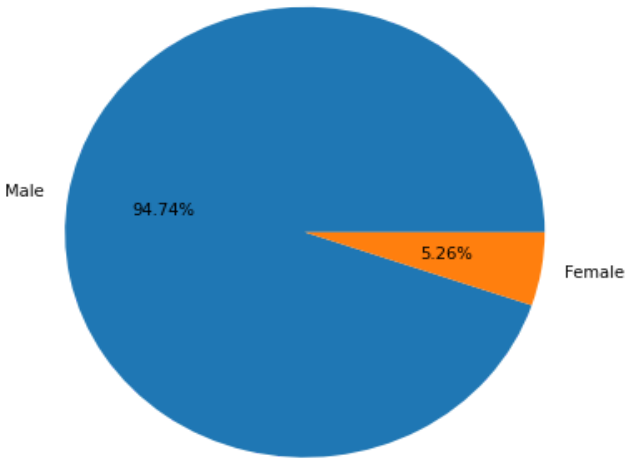
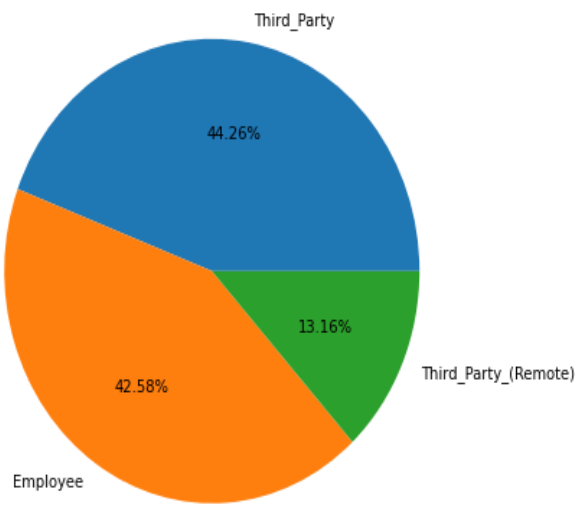

```
[16] print(df[df['Industry Sector'] == 'Mining'].Country.value_counts())
     print(df[df['Industry Sector'] == 'Mining'].Local.value_counts())

     Country_01    203
     Country_02     38
     Name: Country, dtype: int64
     Local_03    90
     Local_01    57
     Local_04    56
     Local_02    24
     Local_07    14
     Name: Local, dtype: int64
```

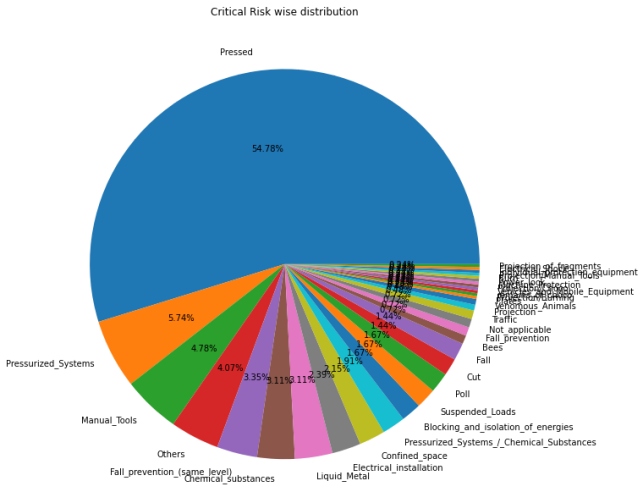
- Most of accidents are belongs to mining sector reported in Local_03 which belongs to Country_01.

Accident level distribution

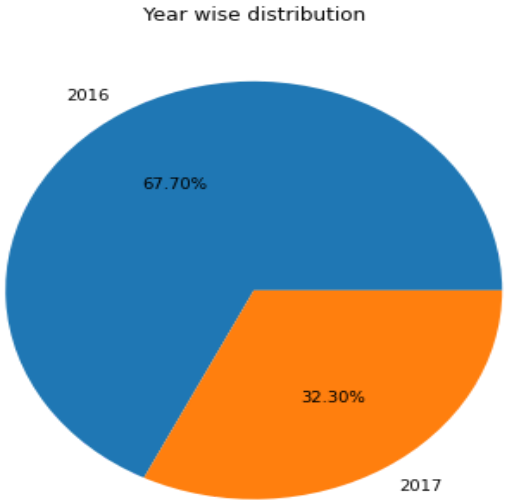


- We have mapped roman nomenclature values of severity level to ordinal.
- Accident level data is skewed with Severity level 1 having most occurrences.
- Severity level 6 data is missing.

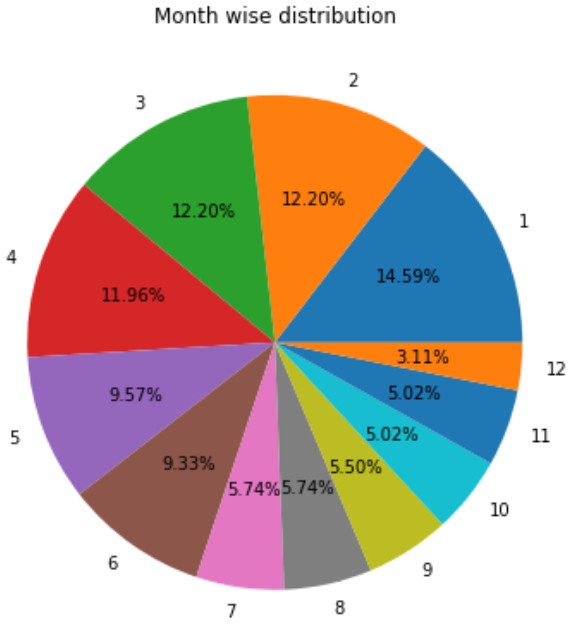
Potential Accident level distribution 	• Potential level 6 data is too less.
Gender wise distribution 	• Proportion of men is significantly higher as per the industry sector.
Employee type wise distribution 	• The large number of Third Party employees including remote types are more prone to accidents based on industry sector.



- Most of the Critical Risks are classified as 'Others', more effort is required to classify this precisely.



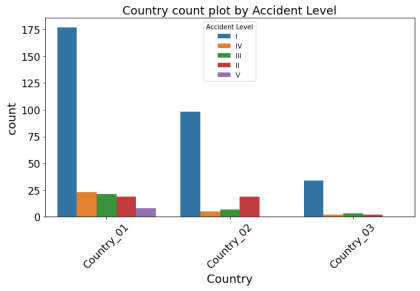
- Security measures are improved as can see 2017 has less accidents reported than 2016.

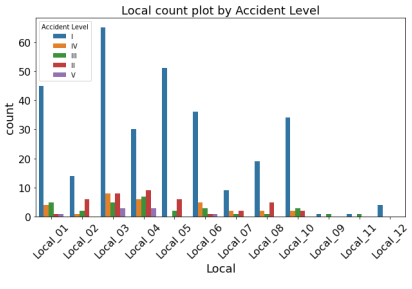


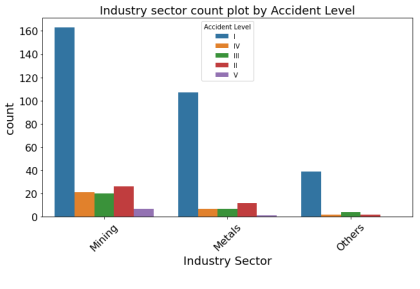
- During the First half of year(month 1-6) more accidents are reported.

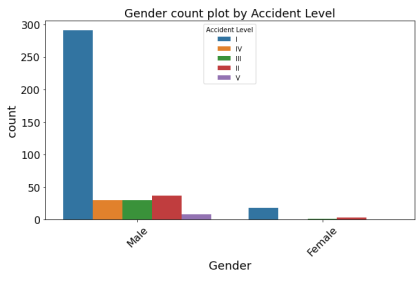
Bi-variate Analysis

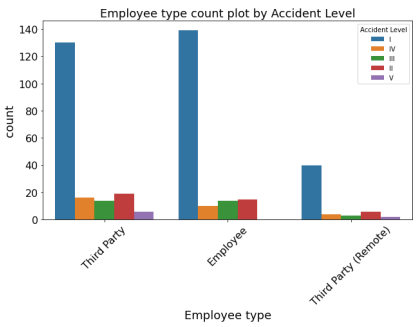
- Count plots by Accident level

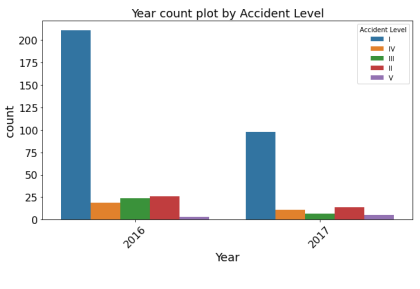


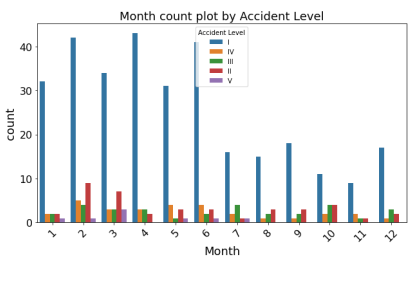


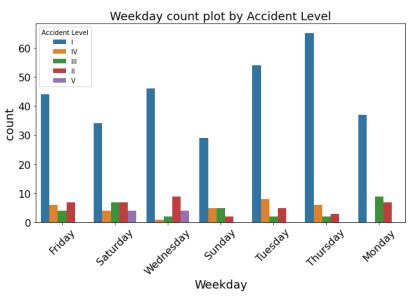


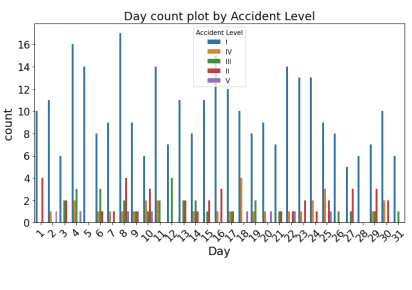








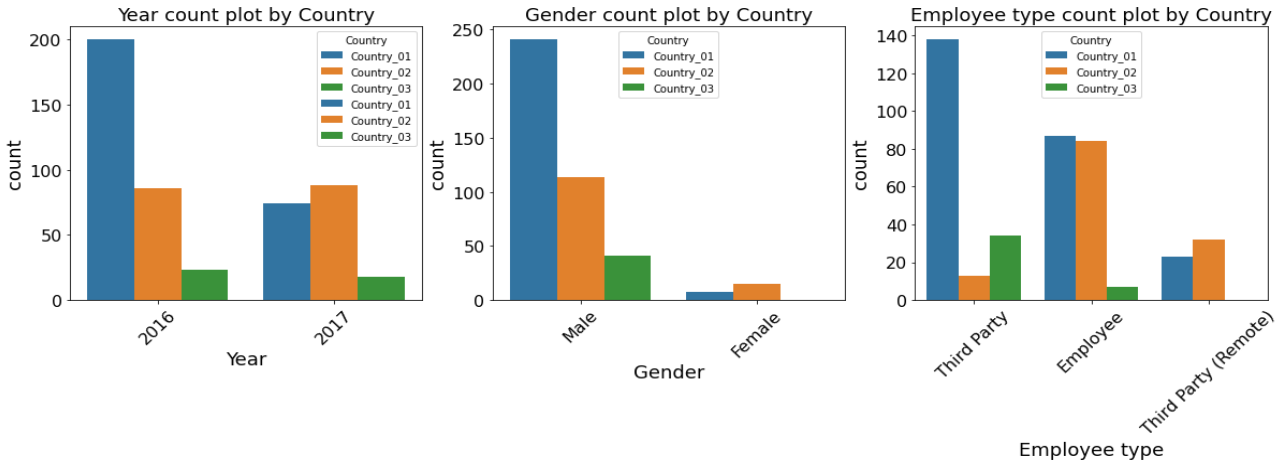




Observations

- ❖ Accident level 1 is reported most for all the countries.
- ❖ Country_01 reported more accidents than country_02 & country_03.
- ❖ Local_03 reported the highest number of accidents.
- ❖ Most of the accidents are from the Mining sector.
- ❖ Males have a higher accident level than females.
- ❖ Severe accident levels (4 & 5) are higher for Third party employee types (including remote).
- ❖ Security measures are improved as can see 2017 has less accidents reported than 2016.
- ❖ It seems that the number of accidents decreased in the latter part of the year / month.
- ❖ The number of accidents increased during the middle of the week and declined since the middle of the week.

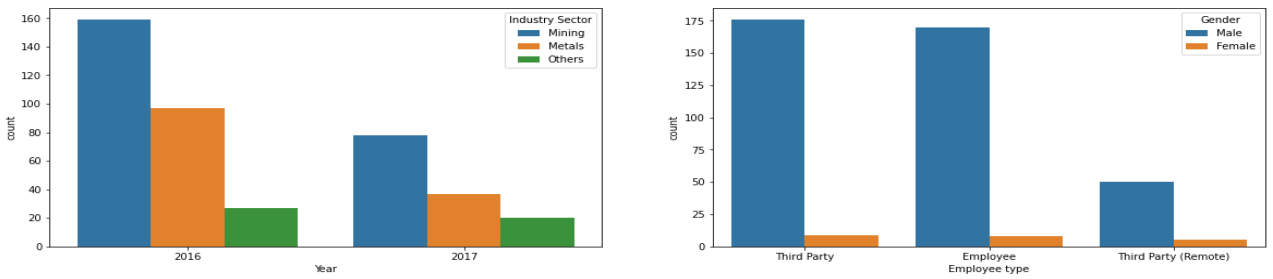
- Count Plot by County



Observations

- ❖ Accidents reported in 2017 are significantly less than 2016.
- ❖ This change in accident count is due to Country_01 as it seems security measures have been improved for Country_01 in 2017.
- ❖ There is no accident reported for female employees in Country_03.
- ❖ There is no accident reported for Third Party(Remote) employee types in Country_03.
- ❖ Third party (including remote) employees reported more accidents than employees.

- Count plot by Industry Sector and Gender



Observations

- ❖ Mining sector reported more accidents which indicates more on-site work for this sector.
- ❖ Males have reported significantly more accidents than females.

NLP Pre-processing

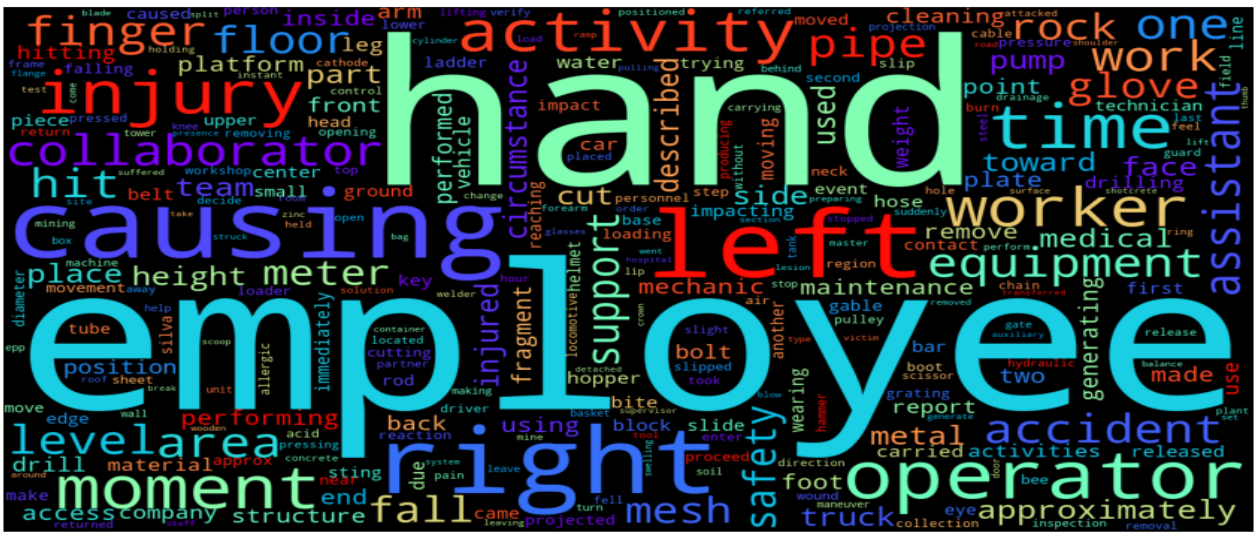
- Taking the "**Description**" column for more analysis and cleaning it by performing below operations and creating a new column called as: "**Cleaned_Description**".

```
============================================================
Converting description to lower case
Replacing apostrophes to the standard lexicons
Removing punctuations
Applying Lemmatizer
Removing multiple spaces between words
Removing stop words
============================================================
```

- As different lines are of different length. We are padding the sequences using the max length and getting maximum and minimum words for the "**Cleaned_Description**" column.

```
Min Desc length: 64
Max Desc length: 680

Min number of words: 10
Max number of words: 98
```

- Generating Word cloud on the "**Cleaned_Description**" column



The wordcloud shows frequency of words in which text size tells relative importance of words in our entire dataset very easily. This can be used where we need to quickly show how people feel about our product in presentation and grabbing attention to the important keywords that we want to represent.

We can see a lot of words related to body, employee, movement, tools and accident. So these words could be used as keywords for chat-bots.

Feature Engineering

Variable Creation

Word embedding is the process by which words are transformed into vectors of real numbers. Most of the algorithms in machine learning cannot process strings or plain text in their raw form. Instead, they require numbers as inputs to be able to function. By transforming words into vectors, word embeddings therefore allow us to process the huge amount of text data and make them fit for machine learning algorithms.

We will use **GloVe and TF-IDF** methods on the "`Cleaned_Description`" column.

1. **GloVe Features:** belongs to the dense vector representations. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.Using glove.6B.200d.txt.

```
glove_df = glove_df.add_suffix('_gl')
glove_df.head()
```

	0_gl	1_gl	2_gl	3_gl	4_gl	5_gl	6_gl	7_gl	8_gl	9_gl	10_gl	11_gl	12_gl	13_gl	14_gl	15_gl	16_gl
0	0.001399	0.089986	0.001088	-0.030902	0.006192	0.068703	-0.102348	-0.028767	-0.062569	0.041729	-0.016129	0.013318	0.112337	0.001674	0.087108	-0.019333	-0.036656
1	-0.014662	0.049947	0.024440	-0.106530	0.004334	0.028683	-0.097660	-0.064637	-0.018871	-0.047895	0.040154	0.049868	0.042412	0.023929	-0.010415	-0.002720	-0.033393
2	-0.008349	-0.058275	-0.094715	-0.074118	-0.024737	0.027817	-0.083538	-0.058431	-0.055072	-0.028999	0.008551	-0.006957	-0.000247	0.096532	0.113783	0.001838	-0.041752
3	0.000628	0.058891	-0.061672	-0.058790	-0.013520	0.062692	-0.055923	-0.049526	-0.003858	0.029369	0.049371	0.025559	0.046074	0.083793	0.077206	-0.015460	0.012461
4	-0.034495	0.023507	-0.062407	-0.061290	-0.002992	0.029277	-0.086362	-0.085732	-0.043758	-0.044062	0.043147	-0.017204	0.026751	0.062899	0.090752	-0.004989	-0.061567

5 rows × 200 columns

2. **TF-IDF Features:** relies on a sparse vector representation. Convert a collection of raw documents to a matrix of TF-IDF features.

```
TFIDF Shape:  (418, 30)
```

	TFIDF_activity	TFIDF_area	TFIDF_causing	TFIDF_employee	TFIDF_hand	TFIDF_injury	TFIDF_left	TFIDF_operator	TFIDF_right	TFIDF_time
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

3. **Label Encoding:** Applying Label encoding on columns: `Weekday, Accident Level, Potential Accident Level`

   LabelEncoder can be used to normalize labels. It can also be used to transform non-numerical labels (as long as they are hashable and comparable) to numerical labels.

4. **One-Hot Encoding:** Applying one-hot encoding on categorical columns: `Accident Level, Country, Local, Gender, Industry Sector, Employee Type, Critical Risk.`

   Created a new dataset called as dummy_target for target variable "`Accident Level`"

Final Shape after applying above feature engineering and creating a merged dataset.

```
Final Shape after applying feature engineering:  (418, 63)
```

	Year	Month	Day	WeekofYear	Weekday	Accident Level	Potential Accident Level	Country_01	Country_02	Country_03	Local_01	Local_02	Local_03	Local_04
0	2016	1	1	53	4	0	3	1	0	0	1	0	0	0
1	2016	1	2	53	5	0	3	0	1	0	0	1	0	0
2	2016	1	6	1	2	0	2	1	0	0	0	0	1	0

Splitting dataset into train and test set

- Creating Train and Test dataset using Label encoding target variable

```
[58] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 1, stratify = y)
```

- Creating Train and Test dataset using One-Hot encoding target variable

```
[59] X_train, X_test, y_train_dummy_target, y_test_dummy_target = train_test_split(X, dummy_target, test_size = 0.20,
                                                                                   random_state = 1, stratify = y)
```

```
print('X_train shape : ({0},{1})'.format(X_train.shape[0], X_train.shape[1]))
print('y_train shape : ({0},)'.format(y_train.shape[0]))
print('X_test shape : ({0},{1})'.format(X_test.shape[0], X_test.shape[1]))
print('y_test shape : ({0},)'.format(y_test.shape[0]))

print('y_train_dummy_target : ({0},)'.format(y_train_dummy_target.shape[0]))
print('y_test_dummy_target shape : ({0},)'.format(y_test_dummy_target.shape[0]))
```

```
X_train shape : (334,111)
y_train shape : (334,)
X_test shape : (84,111)
y_test shape : (84,)
y_train_dummy_target : (334,)
y_test_dummy_target shape : (84,)
```

```
[61] y_train.value_counts()
```

```
0    247
1     32
2     25
3     24
4      6
Name: Accident Level, dtype: int64
```

- The target variable is not balanced so there is a need to apply an upsampling technique. Hence applying SMOTE to balance the dataset.

Upsampling Data: SMOTE

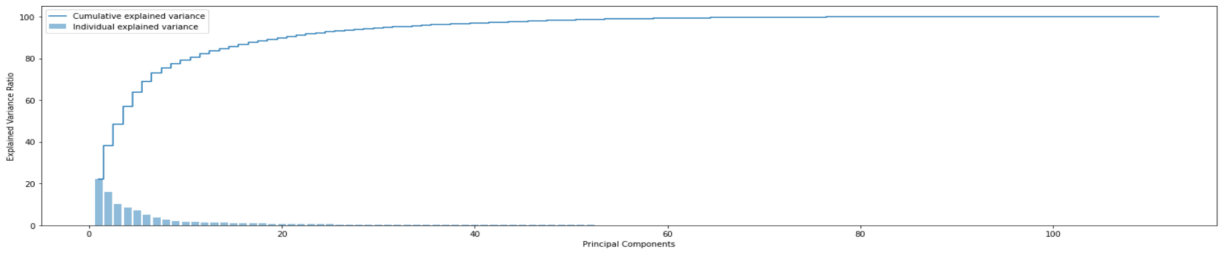
```
sm = SMOTE(random_state=1)
X_train_smote, y_train_smote = sm.fit_resample(X_train, y_train)
df_smote = pd.concat([pd.DataFrame(X_train_smote), pd.DataFrame(y_train_smote)], axis=1)
# df_smote.columns = [f_df.drop(['Accident Level'], axis = 1).columns]
```

```
[63] # Separate input features and target
     X_train_smote = pd.DataFrame(df_smote.drop(['Accident Level'], axis=1)) # Considering all Predictors
     y_train_smote = pd.DataFrame(df_smote['Accident Level'])
```

```
# Display new accident level counts
y_train_smote['Accident Level'].value_counts()
```

```
4    247
3    247
2    247
1    247
0    247
Name: Accident Level, dtype: int64
```

Univariate analysis: PCA




```
pca = PCA(n_components = 0.90)
X_train_reduced = pca.fit_transform(X_train)
X_test_reduced = pca.transform(X_test)
```

```
print(X_train.shape)
print(X_train_smote.shape)
print(X_train_reduced.shape)
```

```
(334, 111)
(1235, 111)
(334, 21)
```

Observation:

- For above PCA we can see that the first 20 features tend to contribute to 90% of accuracy.

Design, Test and Train Machine Unsupervised learning classifiers

We selected below classifier models to run on all three dataset created:

1. Logistic Regression
2. Ridge
3. KNeighborsClassifier
4. SVC
5. RandomForestClassifier
6. BaggingClassifier
7. ExtraTreesClassifier
8. AdaBoostClassifier
9. GradientBoostingClassifier
10. LGBMClassifier
11. XGBClassifier

- **Running models using label encoded train and test dataset**

	Method	Train Accuracy	Test Accuracy	Precision	Recall	F1-Score	Multi-Class Logloss
1	LogisticRegression	0.760479	0.714286	0.553571	0.714286	0.623742	0.895054
2	RidgeClassifier	0.760479	0.726190	0.555850	0.726190	0.629704	1.000000
3	KNeighborsClassifier	0.766467	0.702381	0.551236	0.702381	0.617697	6.886016
4	SVC	0.739521	0.738095	0.544785	0.738095	0.626875	0.882883
5	DecisionTreeClassifier	0.997006	0.595238	0.595238	0.595238	0.594998	13.979981
6	RandomForestClassifier	0.961078	0.726190	0.542456	0.726190	0.621018	3.470738
7	BaggingClassifier	0.964072	0.726190	0.549071	0.726190	0.625331	1.307009
8	ExtraTreesClassifier	0.997006	0.738095	0.551348	0.738095	0.631199	0.862198
9	AdaBoostClassifier	0.745509	0.714286	0.540070	0.714286	0.615079	1.264493
10	GradientBoostingClassifier	0.985030	0.726190	0.603480	0.726190	0.650510	0.929764
11	LGBMClassifier	0.997006	0.690476	0.563283	0.690476	0.620428	1.359473
12	XGBClassifier	0.973054	0.726190	0.555850	0.726190	0.629704	0.940114

Observation:

- ❖ F1 Score for Logistic Regression, Ridge Classifier, Gradient Boosting Classifier is higher.
- ❖ By comparing the results from all above methods, we can select the best method as Gradient Boosting Classifier with f1-score 65.05%

- **Running models using train and test dataset after applying SMOTE**

	Method	Train Accuracy	Test Accuracy	Precision	Recall	F1-Score	Multi-Class Logloss
1	LogisticRegression	0.432389	0.309524	0.616790	0.309524	0.397904	1.527526
2	RidgeClassifier	0.866397	0.071429	0.005102	0.071429	0.009524	1.000000
3	KNeighborsClassifier	0.908502	0.142857	0.474589	0.142857	0.150594	9.354805
4	SVC	0.295547	0.071429	0.005102	0.071429	0.009524	1.601678
5	DecisionTreeClassifier	0.999190	0.500000	0.578755	0.500000	0.533924	17.269388
6	RandomForestClassifier	0.998381	0.607143	0.592472	0.607143	0.598975	1.836841
7	BaggingClassifier	0.998381	0.595238	0.595448	0.595238	0.594215	2.290931
8	ExtraTreesClassifier	0.999190	0.654762	0.593197	0.654762	0.619769	0.975121
9	AdaBoostClassifier	0.636437	0.642857	0.621212	0.642857	0.629181	1.333566
10	GradientBoostingClassifier	0.995142	0.666667	0.594830	0.666667	0.628178	1.118899
11	LGBMClassifier	0.999190	0.714286	0.560579	0.714286	0.628166	1.478114
12	XGBClassifier	0.999190	0.714286	0.625850	0.714286	0.666486	1.065537

Observation:

- ❖ From the results shown above, all methods are over-fitting the training data for the SMOTE dataset.

- **Running models using train and test dataset after applying PCA**

	Method	Train Accuracy	Test Accuracy	Precision	Recall	F1-Score	Multi-Class Logloss
1	LogisticRegression	0.736527	0.738095	0.551348	0.738095	0.631199	0.868395
2	RidgeClassifier	0.739521	0.738095	0.551348	0.738095	0.631199	1.000000
3	KNeighborsClassifier	0.754491	0.702381	0.558303	0.702381	0.622109	6.882591
4	SVC	0.739521	0.738095	0.544785	0.738095	0.626875	0.892725
5	DecisionTreeClassifier	0.997006	0.583333	0.640564	0.583333	0.606853	14.391157
6	RandomForestClassifier	0.973054	0.738095	0.551348	0.738095	0.631199	3.892421
7	BaggingClassifier	0.967066	0.738095	0.551348	0.738095	0.631199	1.702225
8	ExtraTreesClassifier	0.997006	0.738095	0.544785	0.738095	0.626875	1.318288
9	AdaBoostClassifier	0.766467	0.738095	0.551348	0.738095	0.631199	1.234522
10	GradientBoostingClassifier	0.979042	0.714286	0.553571	0.714286	0.623742	1.106259
11	LGBMClassifier	0.997006	0.702381	0.551236	0.702381	0.617697	1.188462
12	XGBClassifier	0.967066	0.702381	0.544345	0.702381	0.613347	0.918235

Observation:

- ❖ F1 Score for Logistic Regression, Ridge Classifier, Gradient Boosting Classifier, AdaBoost is higher.
- ❖ Multiclass log-loss for SVC is lowest with F1 Score of 62.21%

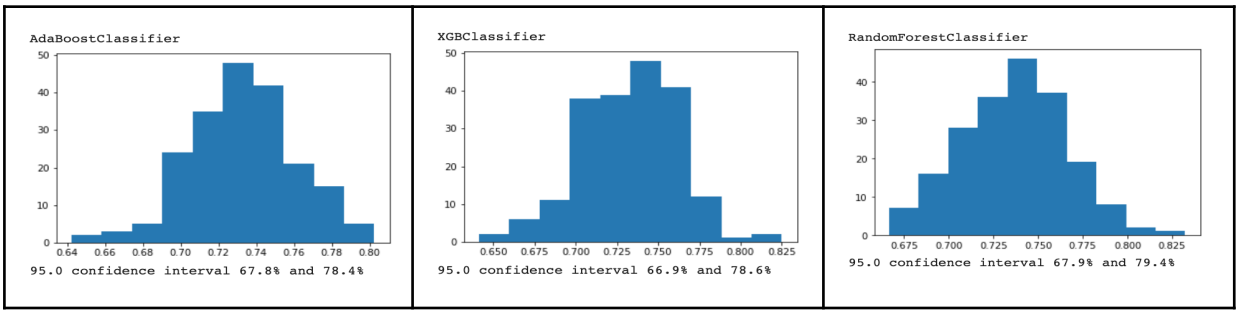
- **Applying Hyperparameter Tuning:**

Due to resource constraint on Google colab we were not able to apply hyper-parameter to all the above models. Following models were able to run showing best possible parameters:

```
LogisticRegression
Best F1_Score: -0.903326 using {'C': 0.01, 'penalty': 'l2', 'solver': 'newton-cg'}
-0.903326 (0.021604) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'newton-cg'}
95% Confidence interval range: (-0.9465 %, -0.8601 %)
Total duration 24.87698459625244

KNeighborsClassifier
Best F1_Score: -2.108952 using {'metric': 'manhattan', 'n_neighbors': 19, 'weights': 'uniform'}
-2.108952 (0.471692) with: {'metric': 'manhattan', 'n_neighbors': 19, 'weights': 'uniform'}
95% Confidence interval range: (-3.0523 %, -1.1656 %)
Total duration 7.477066278457642

SVC
Best F1_Score: 0.000000 using {'C': 50, 'decision_function_shape': 'ovo', 'gamma': 'scale', 'kernel': 'poly'}
0.000000 (0.000000) with: {'C': 50, 'decision_function_shape': 'ovo', 'gamma': 'scale', 'kernel': 'poly'}
95% Confidence interval range: (0.0000 %, 0.0000 %)
Total duration 9.680494785308838

RandomForestClassifier
Best F1_Score: -0.902281 using {'max_features': 'log2', 'n_estimators': 1000}
-0.902281 (0.029312) with: {'max_features': 'log2', 'n_estimators': 1000}
95% Confidence interval range: (-0.9609 %, -0.8437 %)
Total duration 125.843674659729

AdaBoostClassifier
Best F1_Score: -1.144518 using {'learning_rate': 0.1, 'n_estimators': 30}
-1.144518 (0.054813) with: {'learning_rate': 0.1, 'n_estimators': 30}
95% Confidence interval range: (-1.2541 %, -1.0349 %)
Total duration 36.00992703437805
```

Applying Bootstrap Sampling

Bootstrap sampling was applied to `RandomForestClassifier, AdaBoostClassifier, XGBClassifier.`

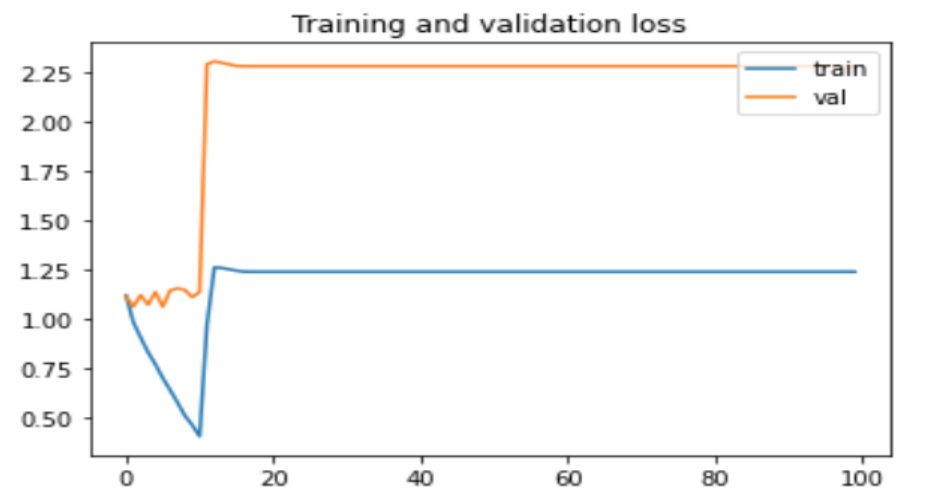


Design, Test and Train Machine Neural networks classifiers

Evaluating the model's accuracy using categorical columns and tf-idf features from accident description and label encoded target variable. We can use simple densely connected neural networks sequential models to make predictions.

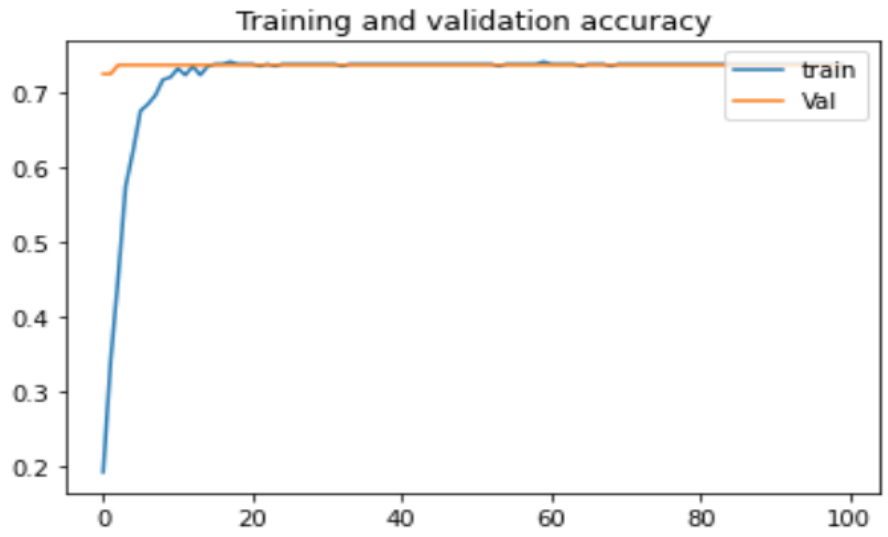
The sequential model is a linear stack of layers. You create a sequential model by calling the keras_model_sequential () function then a series of layer functions: Note that Keras objects are modified in place which is why it's not necessary for the model to be assigned back to after the layers are added.

The output here is shown below:

<table>
<tr><td colspan="2">1. Running model using categorical columns and tf-idf features from accident description and label encoded target variable.</td></tr>
<tr><td>Model Summary</td><td>
Model: "sequential"
_____
Layer (type) Output Shape Param #
=====
dense (Dense) (None, 50) 5600
dense_1 (Dense) (None, 100) 5100
dense_2 (Dense) (None, 150) 15150
dense_3 (Dense) (None, 40) 6040
dense_4 (Dense) (None, 1) 41
=====
Total params: 31,931
Trainable params: 31,931
Non-trainable params: 0
_____
</td></tr>
<tr><td>Model Accuracy</td><td>72.62</td></tr>
<tr><td>Training and Validation Loss</td><td>
Text(0.5, 1.0, 'Training and validation loss')


</td></tr>
<tr><td>Observation</td><td>It can be identified from the learning curve of the training loss only. It is showing noisy values of relatively high loss, indicating that the model was unable to learn the training dataset at all and</td></tr>
</table>

	the model does not have a suitable capacity for the complexity of the dataset.
2. Running model using **categorical columns** and **tf-idf features** from accident description and **One-Hot encoded target variable**.	
Model Summary	Model: "sequential_1" _____ Layer (type) Output Shape Param # ===== dense_5 (Dense) (None, 10) 1120 dropout (Dropout) (None, 10) 0 batch_normalization (BatchN ormalization) (None, 10) 40 dense_6 (Dense) (None, 10) 110 dropout_1 (Dropout) (None, 10) 0 batch_normalization_1 (Batc hNormalization) (None, 10) 40 dense_7 (Dense) (None, 5) 55 ===== Total params: 1,365 Trainable params: 1,325 Non-trainable params: 40 _____
Model Accuracy	Train accuracy: 73.95 Test accuracy: 73.81 Accuracy: 0.738095 Precision: 0.738095 Recall: 0.738095 F1 score: 0.738095
Training and Validation Loss	Text(0.5, 1.0, 'Training and validation loss')  • It is observed from above graph by training and validation loss is decreasing to a point of stability with a minimal gap between the two final loss values

Training and Validation Accuracy	Text(0.5, 1.0, 'Training and validation accuracy')  • We see the learning curves for accuracy on the test dataset plateau, indicating that the model has no longer overfit the training dataset and it is a generalized model.
Observation	One hot encoded: achieved a test accuracy of 73.81% and f1-score of 73.81% with original data + TF-IDF features from accident description column.
3. Running model using **categorical columns** and **tf-idf features** from accident description and **One-Hot encoded target variable** using **SMOTE** data.	
Model Summary	Model: "sequential_2" _____ Layer (type) Output Shape Param # ===== dense_8 (Dense) (None, 10) 1120 dropout_2 (Dropout) (None, 10) 0 batch_normalization_2 (Batc (None, 10) 40 hNormalization) dense_9 (Dense) (None, 10) 110 dropout_3 (Dropout) (None, 10) 0 batch_normalization_3 (Batc (None, 10) 40 hNormalization) dense_10 (Dense) (None, 5) 55 ===== Total params: 1,365 Trainable params: 1,325 Non-trainable params: 40 _____
Model Accuracy	Train accuracy: 20.00 Test accuracy: 8.33 Accuracy: 0.035714 Precision: 0.063830 Recall: 0.035714 F1 score: 0.045802

Training and Validation Loss	Text(0.5, 1.0, 'Training and validation loss')  • It is observed from above graph by training and validation loss is increasing and gap is increasing between the two final loss values
Training and Validation Accuracy	Text(0.5, 1.0, 'Training and validation accuracy')  • We see the learning curves for accuracy are not converging at all.
Observation	Based on the Train and Validation accuracy and loss graphs it is observed that the model with SMOTE data is highly overfitting. Hence applying SMOTE is not a recommended approach.

Design, train and test LSTM classifiers

Data Pre-processing

- Creating a classification model that uses `'Cleaned_Description' and "Accident Level" as target` columns alone.
- Dividing data into Train and Test set:

```
# Divi                                                    ining sets:
X_text                                                    , y_text_test = train_test_split(X_text, y_text, test_size = 0.20, random_state = 1, stratify = y_text)
print('X_text_train shape : ', X_text_train.shape)
print('y_text_train shape : ', y_text_train.shape)
print('X_text_test shape : ', X_text_test.shape)
print('y_text_test shape : ', y_text_test.shape)
```

ndarray: X_text_train

ndarray with shape (334, 100)

```
X_text_train shape :  (334,)
y_text_train shape :  (334,)
X_text_test shape :  (84,)
y_text_test shape :  (84,)
```

- Converting target train and test data to categorical and applying
- Applying Tokenizer on X train and test data and also creating word embeddings using Glove word embeddings.

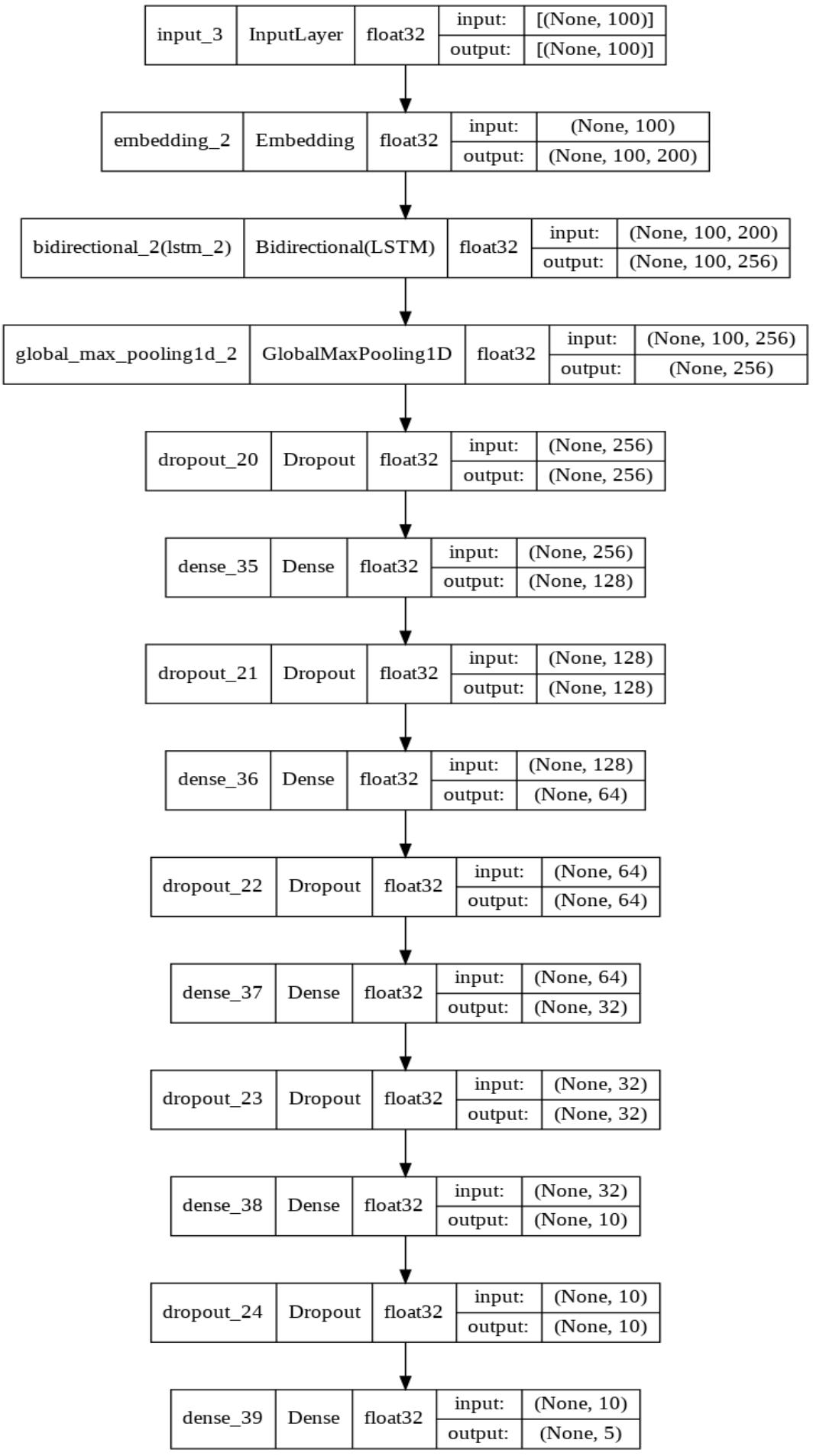
Defining LSTM model

We are using Bi-directional LSTM model using glove word embeddings with 4 dense layers and SGD as an optimizer.

Model Summary

```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 100)]             0

 embedding_2 (Embedding)     (None, 100, 200)          612200

 bidirectional_2 (Bidirectio  (None, 100, 256)         336896
 nal)

 global_max_pooling1d_2 (Glo  (None, 256)              0
 balMaxPooling1D)

 dropout_20 (Dropout)        (None, 256)               0

 dense_35 (Dense)            (None, 128)               32896

 dropout_21 (Dropout)        (None, 128)               0

 dense_36 (Dense)            (None, 64)                8256

 dropout_22 (Dropout)        (None, 64)                0

 dense_37 (Dense)            (None, 32)                2080

 dropout_23 (Dropout)        (None, 32)                0

 dense_38 (Dense)            (None, 10)                330

 dropout_24 (Dropout)        (None, 10)                0

 dense_39 (Dense)            (None, 5)                 55

=================================================================
Total params: 992,713
Trainable params: 380,513
Non-trainable params: 612,200
_________________________________________________________________
```

Model Plot

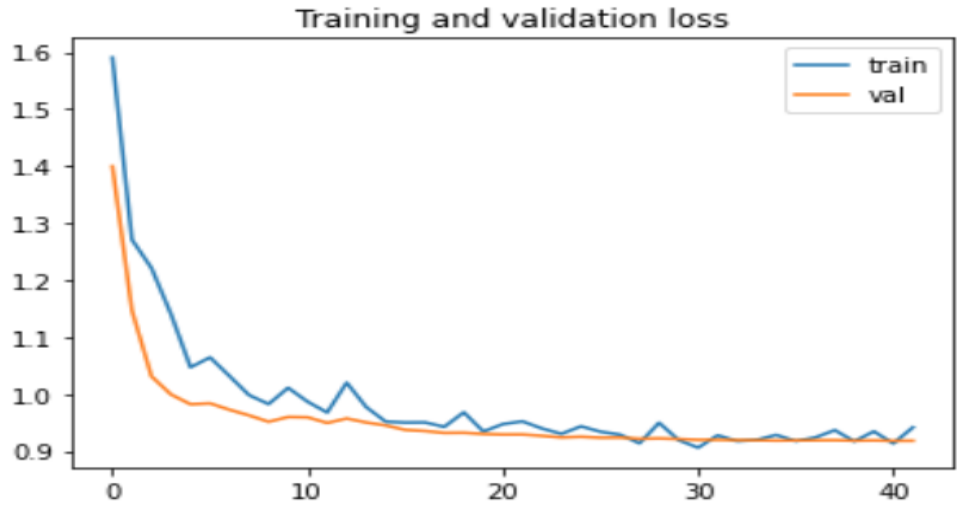

input_3 InputLayer float32 input: [(None, 100)] output: [(None, 100)]
embedding_2 Embedding float32 input: (None, 100) output: (None, 100, 200)
bidirectional_2(lstm_2) Bidirectional(LSTM) float32 input: (None, 100, 200) output: (None, 100, 256)
global_max_pooling1d_2 GlobalMaxPooling1D float32 input: (None, 100, 256) output: (None, 256)
dropout_20 Dropout float32 input: (None, 256) output: (None, 256)
dense_35 Dense float32 input: (None, 256) output: (None, 128)
dropout_21 Dropout float32 input: (None, 128) output: (None, 128)
dense_36 Dense float32 input: (None, 128) output: (None, 64)
dropout_22 Dropout float32 input: (None, 64) output: (None, 64)
dense_37 Dense float32 input: (None, 64) output: (None, 32)
dropout_23 Dropout float32 input: (None, 32) output: (None, 32)
dense_38 Dense float32 input: (None, 32) output: (None, 10)
dropout_24 Dropout float32 input: (None, 10) output: (None, 10)
dense_39 Dense float32 input: (None, 10) output: (None, 5)

<table>
<tr><td>Model Accuracy</td><td>

```
Train accuracy: 73.95
Test accuracy: 73.81
Accuracy: 0.738095
Precision: 0.738095
Recall: 0.738095
F1 score: 0.738095
```

</td></tr>
<tr><td>Training and Validation Loss</td><td>

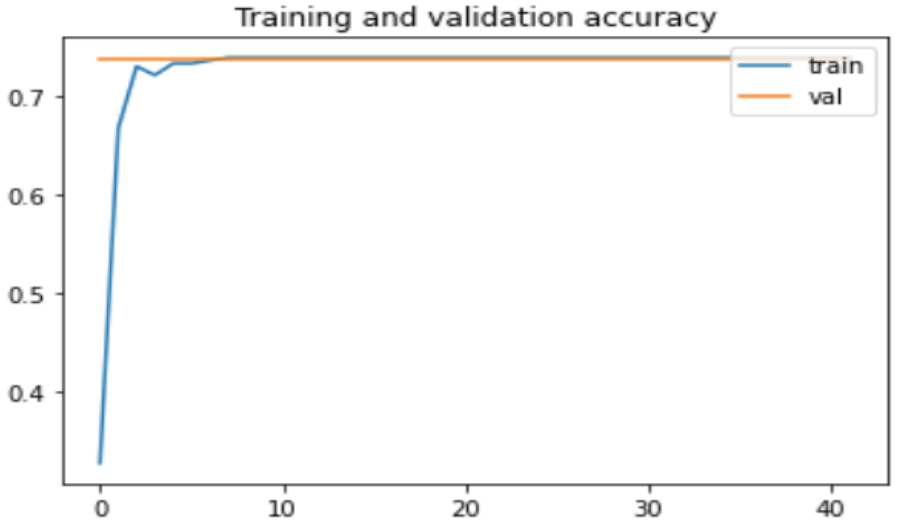
```
Text(0.5, 1.0, 'Training and validation loss')
```



- It is observed from the above graph by training and validation loss is decreasing to a point of stability with a minimal gap between the two final loss values.

</td></tr>
<tr><td>Training and Validation Accuracy</td><td>

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



- We see the learning curves for accuracy on the test dataset plateau, indicating that the model has no longer overfit the training dataset and it is a generalized model.

</td></tr>
<tr><td>Observation</td><td>Based on the Train and Validation accuracy and loss graphs it is observed that the model is giving accuracy of approx 73% with F1 score of 73%.</td></tr>
</table>

Result Summary for Neural network models:

	Method	Accuracy	Precision	Recall	F1-Score
0	label encoded model	0.345238	0.345238	0.345238	0.345238
1	One-Hot encoded model	0.738095	0.738095	0.738095	0.738095
2	One-Hot encoded with SMOTE	0.035714	0.066667	0.035714	0.046512
3	LSTM-Bidirectional	0.738095	0.738095	0.738095	0.738095

Overall Summary

1. Able to predict the accident level with a test accuracy of 73.81% and f1-score of 73.81%
2. We have seven duplicate values in this dataset and dropped those duplicate values.
3. We have no missing values in this dataset.
4. Created new features such as Day, Month, Year, Weekday, weekofyear from the Date column.
5. Target variable – '**Accident Level'** distribution is imbalance.
6. Class imbalance issue is handled using SMOTE: Generate synthetic samples and found out that, for this particular dataset, with original data we have achieved the better results.
7. By comparing the results from all ML methods with original data, we can select the best method as Gradient Boosting Classifier with f1-score 65% with original data.
8. Bootstrap sampling with RandomForestClassifier model with an accuracy range of 67.9% - 77.4% is our best model.
9. Explored below options in Neural Networks:
   a. Convert Classification to Numerical problem: achieved a test accuracy of 72.62%.
   b. Multiclass classification - Target variable - One hot encoded: achieved a test accuracy of 73.81% and f1-score of 73.81% with original data + TF-IDF features from "**Cleaned Description**" column.
   c. Create a model with Text inputs (accident description alone) only: surprisingly achieved a test accuracy of 73.81% and f1-score of 73.81% with original data.
10. **Finally, a bidirectional LSTM model can be considered to productionalized the model and predict the accident level.**

4. Visualization

UI Interactive Board chatbot

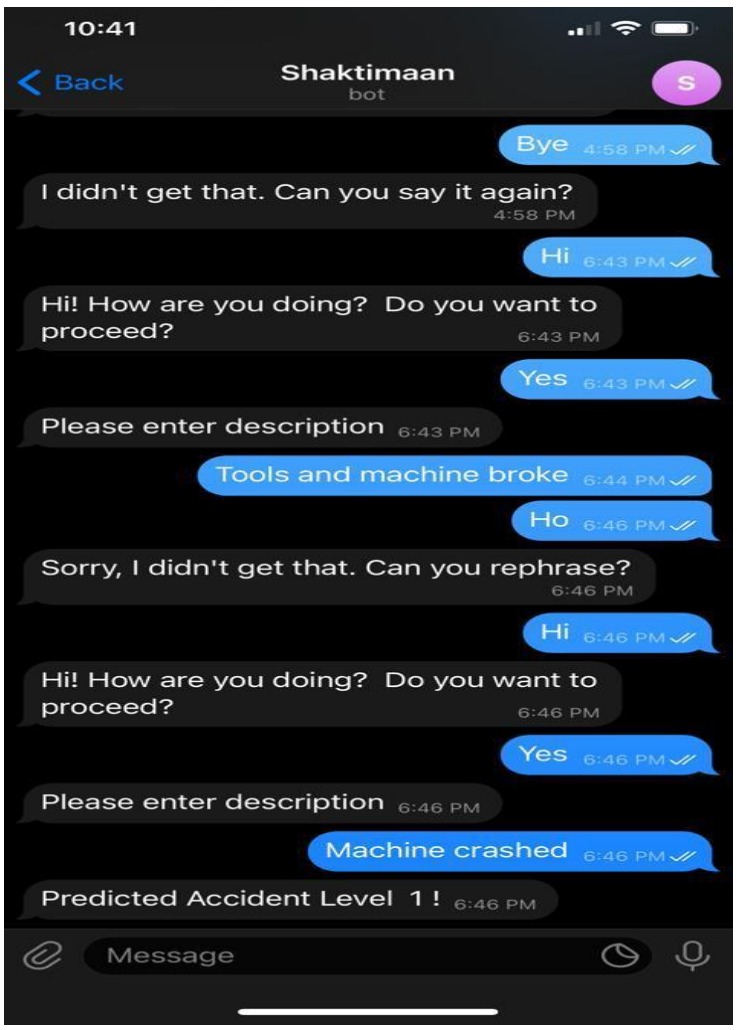
In the industrial design field of human–computer interaction, a user interface (UI) is the space where interactions between humans and machines occur. The goal of this interaction is to allow effective operation and control of the machine from the human end, whilst the machine simultaneously feeds back information that aids the operators' decision-making process.

Design a clickable UI based chatbot interface

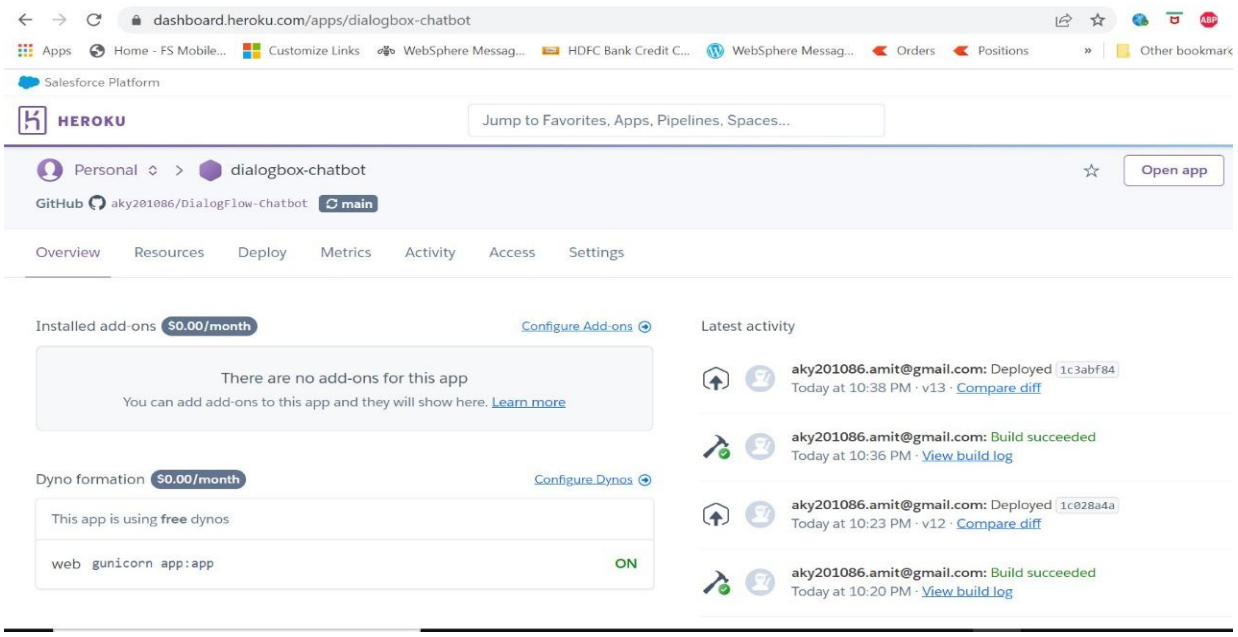
We have performed following activities:

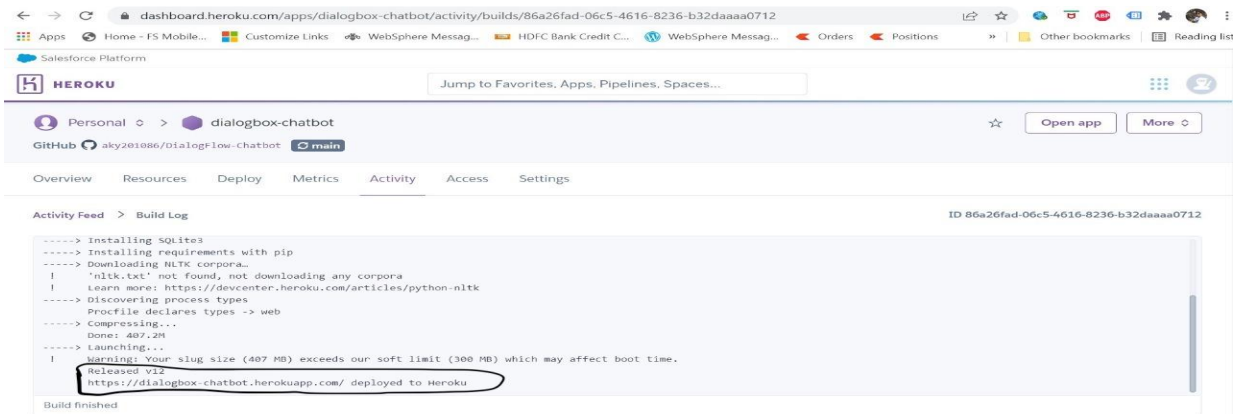
- Pickled the final LSTM model and pushed to Github.
- Created a flask application which exposes Rest Api which takes the input, pre-process that by applying morphological analysis, interacts with the model and returns the predicted Accident level in response.
- Used Google Dialog-flow to design and integrate a conversational user interface and integrated with Telegram to provide user interface.

- Telegram Chat Window

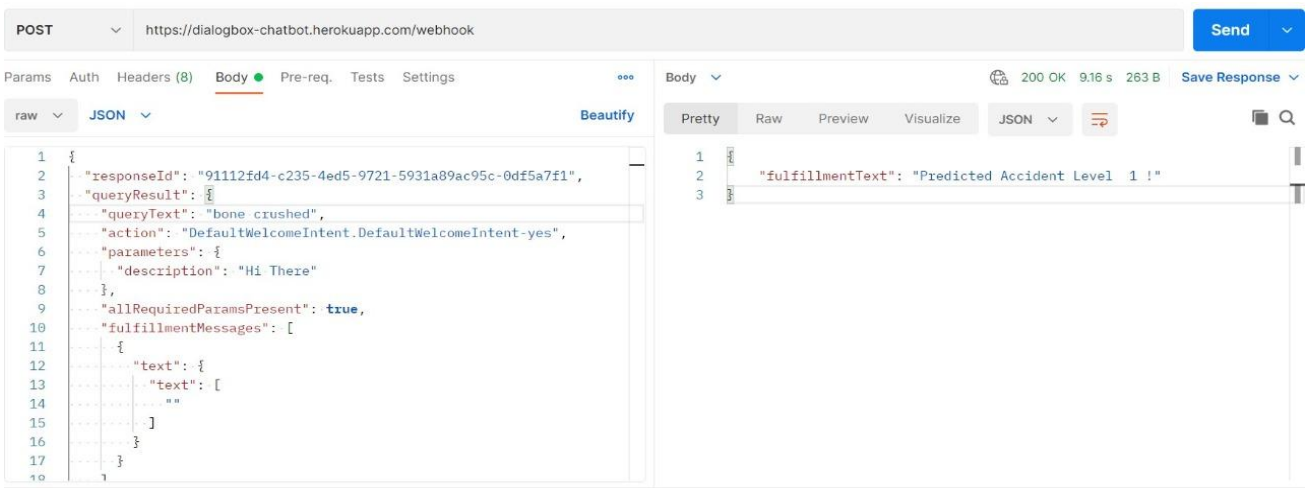


- Flask api deployment on Heroku along with the model.





- Postman api call response logs



5. Limitations

- With more detailed information such as machining data(ex. CNC, Current, Voltage) in plants, weather information, employee's personal data(ex. age, experience in the industry sector, work performance), we can clarify the cause of accidents more correctly.
- At present the model always returns some prediction even if non-relevant accident description is passed as input.

6. Closing Reflections

In this project, we discovered that the main causes of accidents are mistakes in hand-operation and time-related factors. To reduce the occurrences of accidents, more stringent safety standards in hand-operation will be needed in a period when many accidents occur.

Github Repo link:

https://github.com/aky201086/DialogFlow-Chatbot

Telegram Chatbot link:

https://web.telegram.org/z/#5105460156