NATURAL LANGUAGE PROCESSING CHATBOT INTERFACE

Final Report Presentation

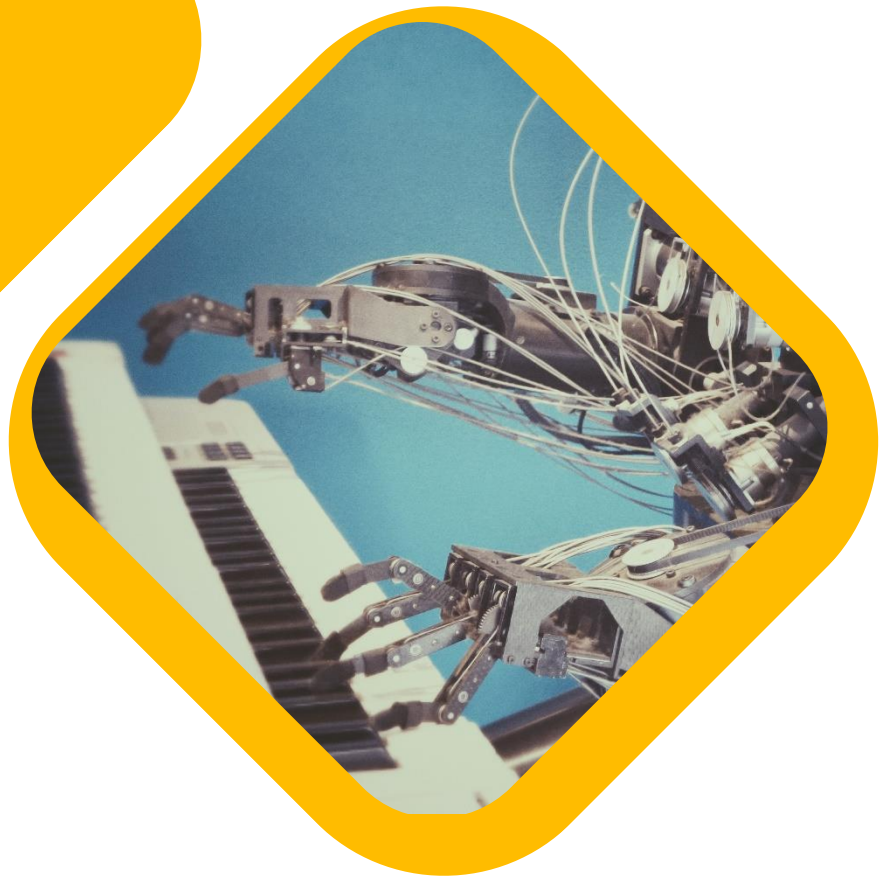
Submitted By

- Shashi Ravuri
- Jata Shankar Poddar
- Abhishek Kumar Singh
- Arjit Singh Matharu
- Vishal
- Shyam Arora

Project Mentor

- Aniket Chhabra

CONTENTS



INTRODUCTION AND PROBLEM STATEMENT

INTRODUCTION

This capstone project is based on designing an ML/DL-based chatbot utility that can help the professionals to highlight the safety risk as per the incident description. In the final report, we need to clean the data and design ML, ANN, RNN and LSTM model and choose the best model for future use.

Problem Statement

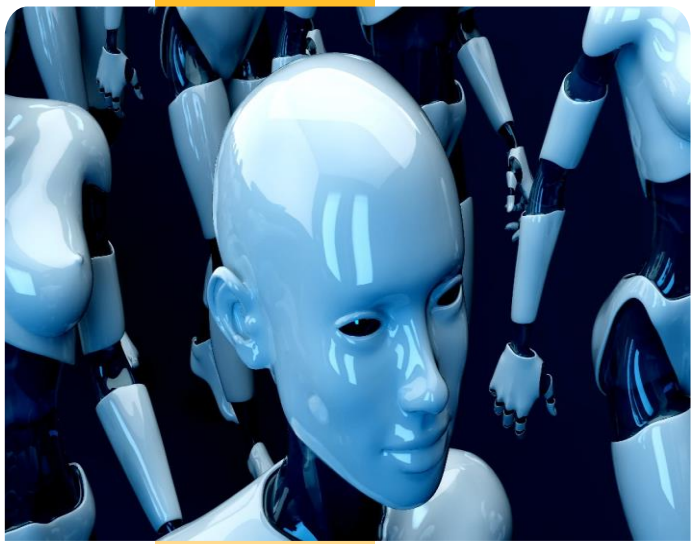
Domain: Industry Safety, NLP-based Chatbot.

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

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

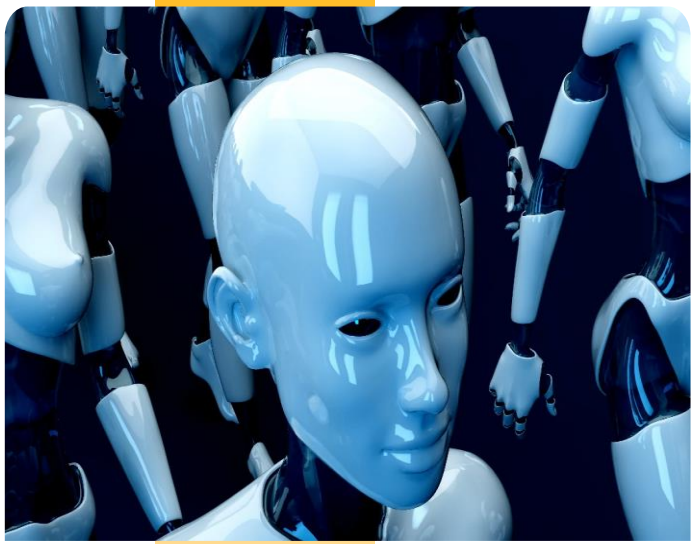
OVERVIEW OF THE FINAL PROCESS



The brief approach for the solution is given as follows:

- The objective defines that the end solution requires for a final report to build model that predicts the Potential Accident Level by providing a description of the accident in text format.
- The first step is data importing, Data Cleansing and Data-Preprocessing to have good cleaned data for the input dataset for the model to predict the expected output.
- Once data cleansing and preprocessing completed, the EDA (exploratory data analysis) has been performed to understand the dataset more deeply and identify trends and patterns which also help us to understand the structure of the dataset.
- After the EDA, the model buildings have been given in which we have built ML-based models, ANN-based models, RNN and LSTM Models. A detailed summary and evaluation of all the models are given as well to compare the results and performance.
- Choose the best model and pickle the model for future use.

OVERVIEW OF MILESTONE-1



The brief approach for the solution is given as follows:

- The objective defines that the end solution requires for an interim report to build ML model that predicts the Potential Accident Level based on the description.
- The first step is data importing where we import the data in python environment to do further analysis.
- In the data cleansing, we rename the incorrect columns name, check the missing value, remove duplicates, check outliers and streamline the date column.
- After the data cleansing, we have done text lowercase, text punctuation, remove punctuation, remove special characters, remove whitespace, remove default stopwords, stemming and lemmatization in the Data-Preprocessing.
- Once data cleansing and preprocessing completed, the EDA (exploratory data analysis) has been performed to understand the dataset more deeply and identify trends and patterns which also help us to understand the structure of the dataset.
- After the EDA, design train and test basic machine learning classifiers.

IMPORT THE DATASET

Data sources :-

- Original dataset link:- https://www.kaggle.com/ihmstefanini/industrial-safety-and-health-analytics-database

Importing data :-

- We Imported the Dataset from the given xlsx file and the size of the dataset is 425 * 11

Now, we can import the data

```
[39] chatbot_database = pd.read_excel("/content/drive/MyDrive/Capstone/Data Set - industrial_safety_and_health_database_with_accidents_description.xlsx")

[32] chatbot_database.shape

(425, 11)
```

DATA CLEANSING

We have cleaned the dataset by implemented below five steps

Re-named column name

- We have renamed the following columns name

Checked Missing Values

- No missing values are present in the dataset

Removed Duplicates

- There are 14 rows where descriptions are same. We have also removed them from the dataset. The shape of the dataset is now (411,10)

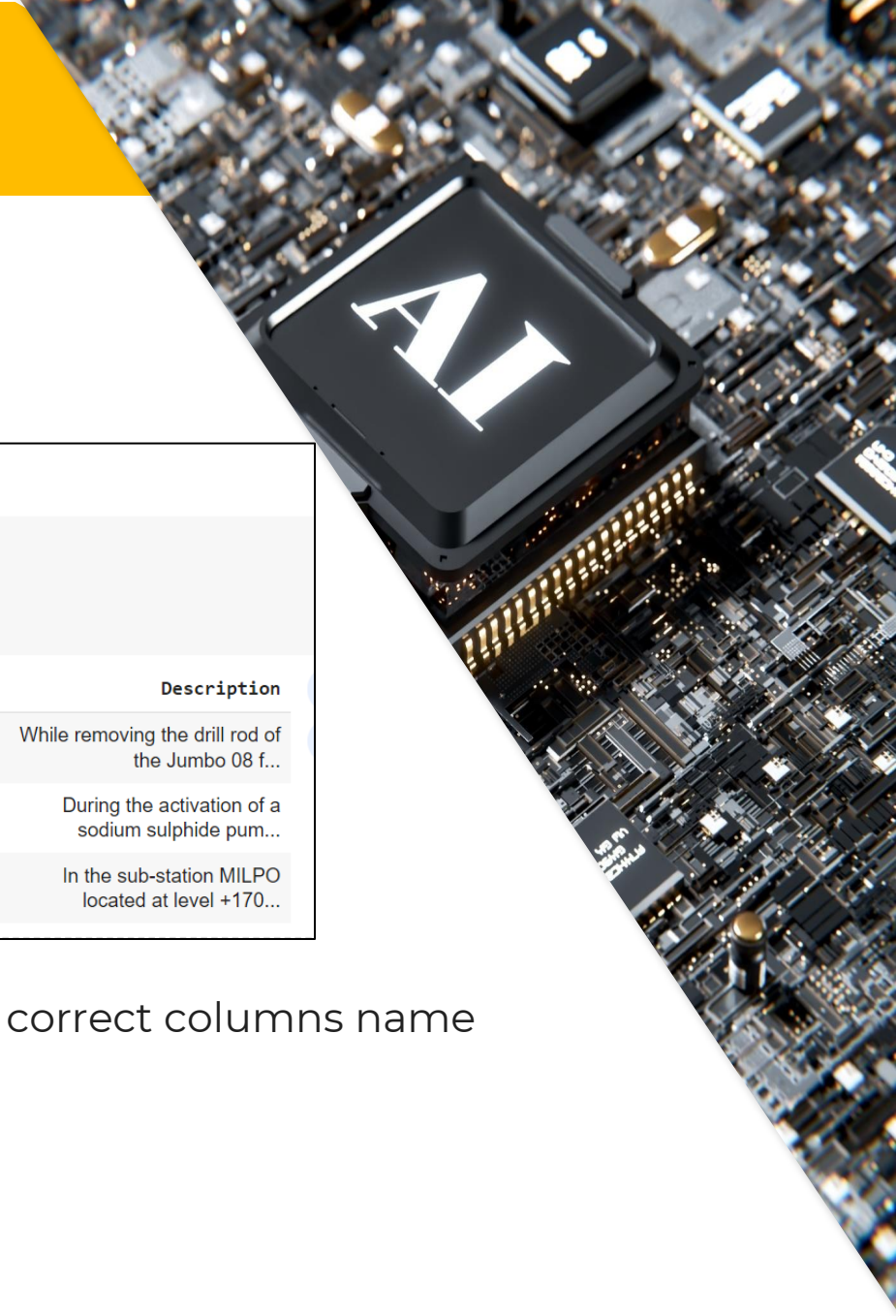
Checked Outliers

- There are no such outliers present in the dataset.

Streamlined date column

- We have streamlined the date column and found that the dataset contains 19 months data from January 2016-July 2017. We have also segregated the months into season to get better understanding of the accidents.

DATA CLEANSING



1. Re-named column name

Executed the below code to correct the columns name

We have 10 columns(features) and 425 rows of data. Let us Rename the column correctly, which is misspelled.

```
[ ] chatbot_database.rename(columns={'Data':'Date', 'Countries':'Country', 'Genre':'Gender', 'Employee or Third Party':'Employee_type',
    'Industry Sector':'Industry_Sector','Accident Level':'Accident_Level','Potential Accident Level':'Potential_Accident_Level',
    'Critical Risk':'Critical_Risk'}, inplace=True)
    chatbot_database.head(3)
```

	Date	Country	Local	Industry_Sector	Accident_Level	Potential_Accident_Level	Gender	Employee_type	Critical_Risk	Description
0	2016-01-01 0:00:00	Country_01	Local_01	Mining	I	IV	Male	Third Party	Pressed	While removing the drill rod of the Jumbo 08 f...
1	2016-01-02 0:00:00	Country_02	Local_02	Mining	I	IV	Male	Employee	Pressurized Systems	During the activation of a sodium sulphide pum...
2	2016-01-06 0:00:00	Country_01	Local_03	Mining	I	III	Male	Third Party (Remote)	Manual Tools	In the sub-station MILPO located at level +170...

Result – Below are the list of columns name which have been replaced by correct columns name

Incorrect column name	Corrected column name
Data	Date
Countries	Country
Genre	Gender
Employee or Third Party	Employee_type
Industry Sector	Industry_Sector
Accident Level	Accident_Level
Potential Accident Level	Potential_Accident_Level
Critical Risk	Critical_Risk

DATA CLEANSING

2. Checking Missing Values

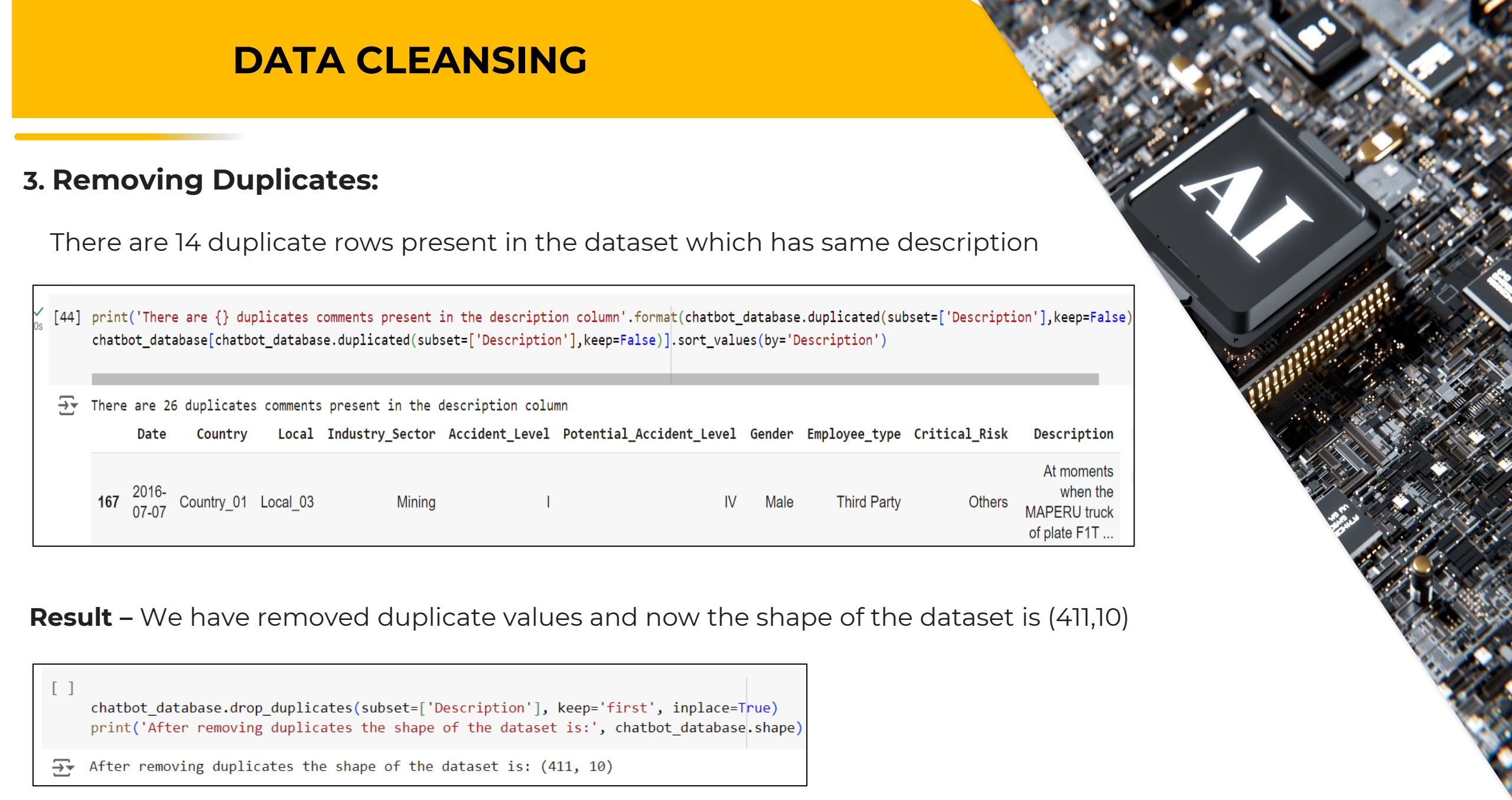
Executed the following code to check if there are any missing values in the dataset:-

```
datatype = pd.DataFrame(chatbot_database.dtypes)
datatype['MissingVal'] = chatbot_database.isnull().sum()
datatype['NUnique']=chatbot_database.nunique()
datatype
```

	0	MissingVal	NUnique
Date	object	0	287
Country	object	0	3
Local	object	0	12
Industry_Sector	object	0	3
Accident_Level	object	0	5
Potential_Accident_Level	object	0	6
Gender	object	0	2
Employee_type	object	0	3
Critical_Risk	object	0	33
Description	object	0	411

Result – There are no such missing value present in the dataset.

DATA CLEANSING



3. Removing Duplicates:

There are 14 duplicate rows present in the dataset which has same description

```
[44] print('There are {} duplicates comments present in the description column'.format(chatbot_database.duplicated(subset=['Description'],keep=False)
     chatbot_database[chatbot_database.duplicated(subset=['Description'],keep=False)].sort_values(by='Description')
```

```
There are 26 duplicates comments present in the description column
```

	Date	Country	Local	Industry_Sector	Accident_Level	Potential_Accident_Level	Gender	Employee_type	Critical_Risk	Description
167	2016-07-07	Country_01	Local_03	Mining	I	IV	Male	Third Party	Others	At moments when the MAPERU truck of plate F1T ...

Result – We have removed duplicate values and now the shape of the dataset is (411,10)

```
[ ]
     chatbot_database.drop_duplicates(subset=['Description'], keep='first', inplace=True)
     print('After removing duplicates the shape of the dataset is:', chatbot_database.shape)
```

```
After removing duplicates the shape of the dataset is: (411, 10)
```

DATA CLEANSING

4. Checking Outliers:

Executed the outliers code to see whether there are any outliers present in the dataset:-

```
[ ]  for x in chatbot_database.columns:
         if x != 'Description' and x != 'Date':
             print('--'*30); print(f'Unique values of "{x}" column'); print('--'*30)
             print(chatbot_database[x].unique())
             print("\n")
```

```
Unique values of "Country" column
------------------------------------------------------------
['Country_01' 'Country_02' 'Country_03']


------------------------------------------------------------
Unique values of "Local" column
------------------------------------------------------------
['Local_01' 'Local_02' 'Local_03' 'Local_04' 'Local_05' 'Local_06'
 'Local_07' 'Local_08' 'Local_10' 'Local_09' 'Local_11' 'Local_12']
```

Result – There are no such outliers present in the dataset. However, in the Potential_Accident_Level, there is only 1 value present in level-VI. Hence, we have replaced the same with level-V

As we can see that there is only 1 level VI data present in the potential accident level, so we can replace the same with level V

```
[ ]  chatbot_database['Potential_Accident_Level'] = chatbot_database['Potential_Accident_Level'].replace('VI', 'V')
```

DATA CLEANSING

5. Streamline date column to get some insights

Executed the following code to see the tenure of the dataset

```
[ ]  print(chatbot_database.Date.min())
     print(chatbot_database.Date.max())

     2016-01-01 0:00:00
     2017-07-09 0:00:00
```

Result – This is the 19 months data which has data from January 2016 till July 2017

Now, execute the following code to get year, month, day, weekday, week of year and season Columns which will help us to get insights in EDA section

```
[ ]  chatbot_database['Date'] = pd.to_datetime(chatbot_database['Date'])
     chatbot_database['Year'] = chatbot_database['Date'].apply(lambda x : x.year)
     chatbot_database['Month'] = chatbot_database['Date'].apply(lambda x : x.month)
     chatbot_database['Day'] = chatbot_database['Date'].apply(lambda x : x.day)
     chatbot_database['Weekday'] = chatbot_database['Date'].apply(lambda x : x.day_name())
     chatbot_database['WeekofYear'] = chatbot_database['Date'].apply(lambda x : x.weekofyear)
     chatbot_database.head(3)
```

```
[ ]  def month2seasons(x):
         if x in [9, 10, 11]:
             season = 'Spring'
         elif x in [12, 1, 2,3]:
             season = 'Summer'
         elif x in [4, 5,6]:
             season = 'Autumn'
         elif x in [7, 8]:
             season = 'Winter'
         return season

[ ]  chatbot_database['Season'] = chatbot_database['Month'].apply(month2seasons)
     chatbot_database.head(3)
```

DATA PREPROCESSING

We need to install the below libraries before data preprocessing:-

Install pyspellchecker library for spell check.

```
[ ]  !pip install contractions
     !pip install pyspellchecker
     verbose= False
```

We have executed the following data preprocessing techniques to streamline dataset :-

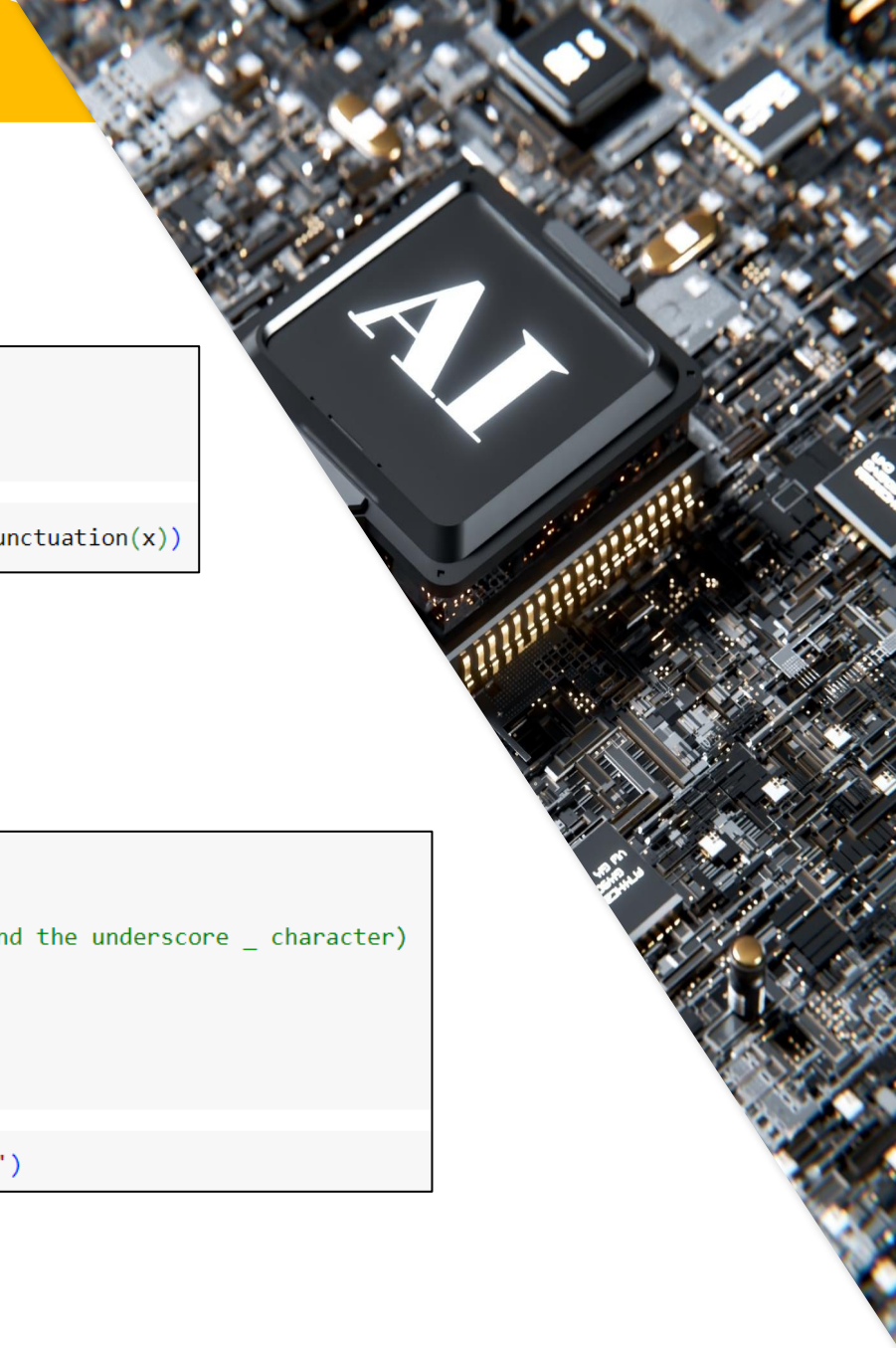
1. Convert into Lower case

```
[ ]  def text_lowercase(text):
         return text.lower()

[ ]  chatbot_database['Description_preprocess'] = chatbot_database['Description'].apply(lambda x: text_lowercase(x))
```

2. Text Punctuation

```
[ ]  import re

     def cleanPunc(sentence):
         cleaned = re.sub(r'[?|!|\'|"|#]',r'',sentence)
         cleaned = re.sub(r'[0-9]',r'',cleaned)
         cleaned = re.sub(r'[.|,|)|(|\|/]',r' ',cleaned)
         cleaned = cleaned.strip()
         cleaned = cleaned.replace("\n"," ")
         return cleaned

[ ]  chatbot_database['Description_preprocess'] = chatbot_database['Description_preprocess'].str.replace("[0-9]", " ")
```

DATA PREPROCESSING



3. Remove Punctuation

```
[ ]  def remove_punctuation(text):
         translator = str.maketrans('', '', string.punctuation)
         return text.translate(translator)
```

```
[ ]  chatbot_database['Description_preprocess'] = chatbot_database['Description_preprocess'].apply(lambda x: remove_punctuation(x))
```

4. Remove Special Characters

```
[ ]  def remove_spc_char(text):
         text.str.replace('[^\w\s]','')
     # \w: Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character)
     # \s: Returns a match where the string contains a white space character.
     # [^]: Returns a match for any character EXCEPT what is written after it.

     # data['result'] = data['result'].map(lambda x: x.lstrip('+-').rstrip('aAbBcC'))
```

```
[ ]  chatbot_database['Description_preprocess'] = chatbot_database['Description_preprocess'].str.replace('[^\w\s]','')
```

DATA PREPROCESSING

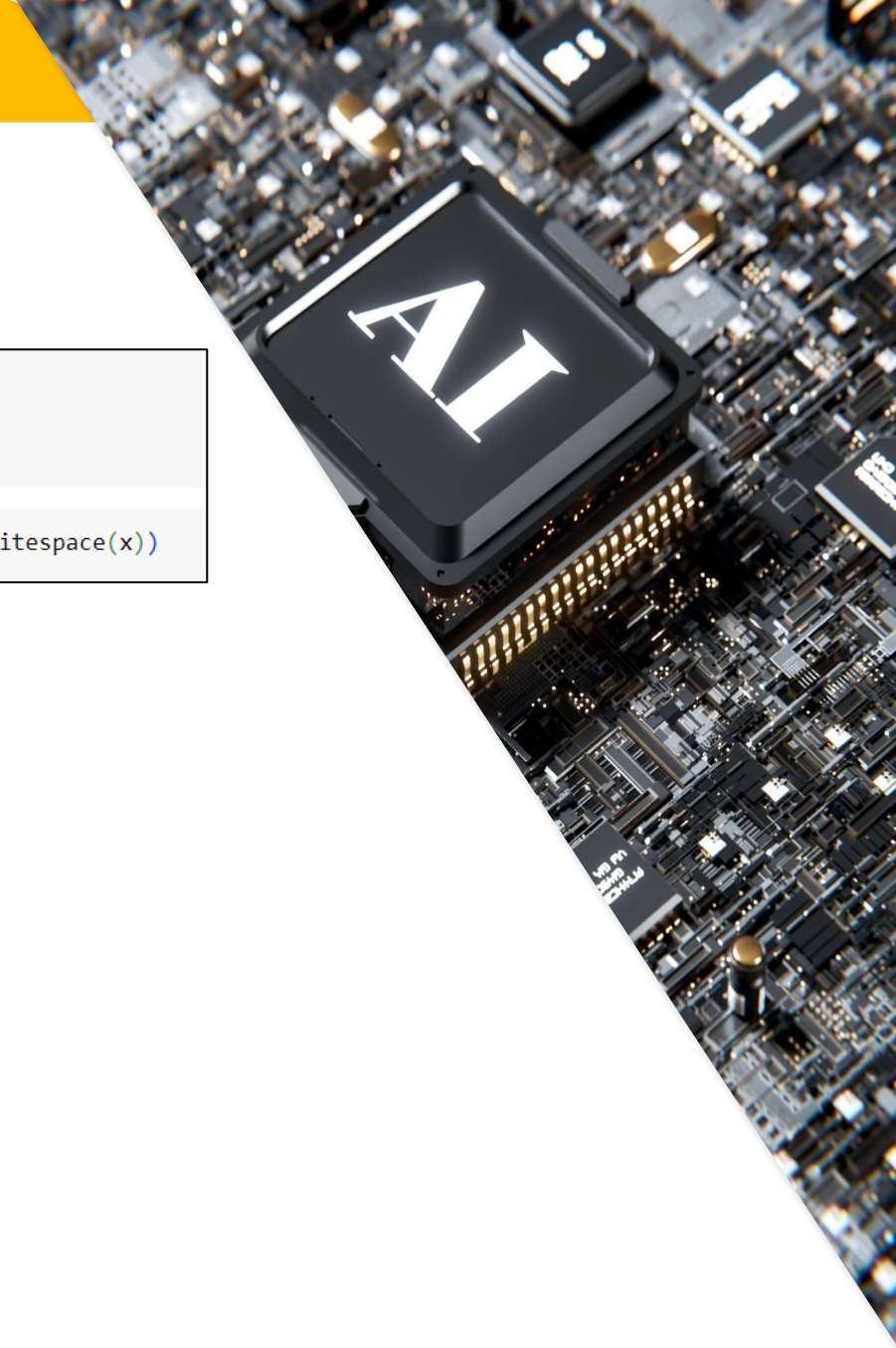
5. Remove whitespaces

```
[ ]  # remove whitespace from text
     def remove_whitespace(text):
         return  " ".join(text.split())
```

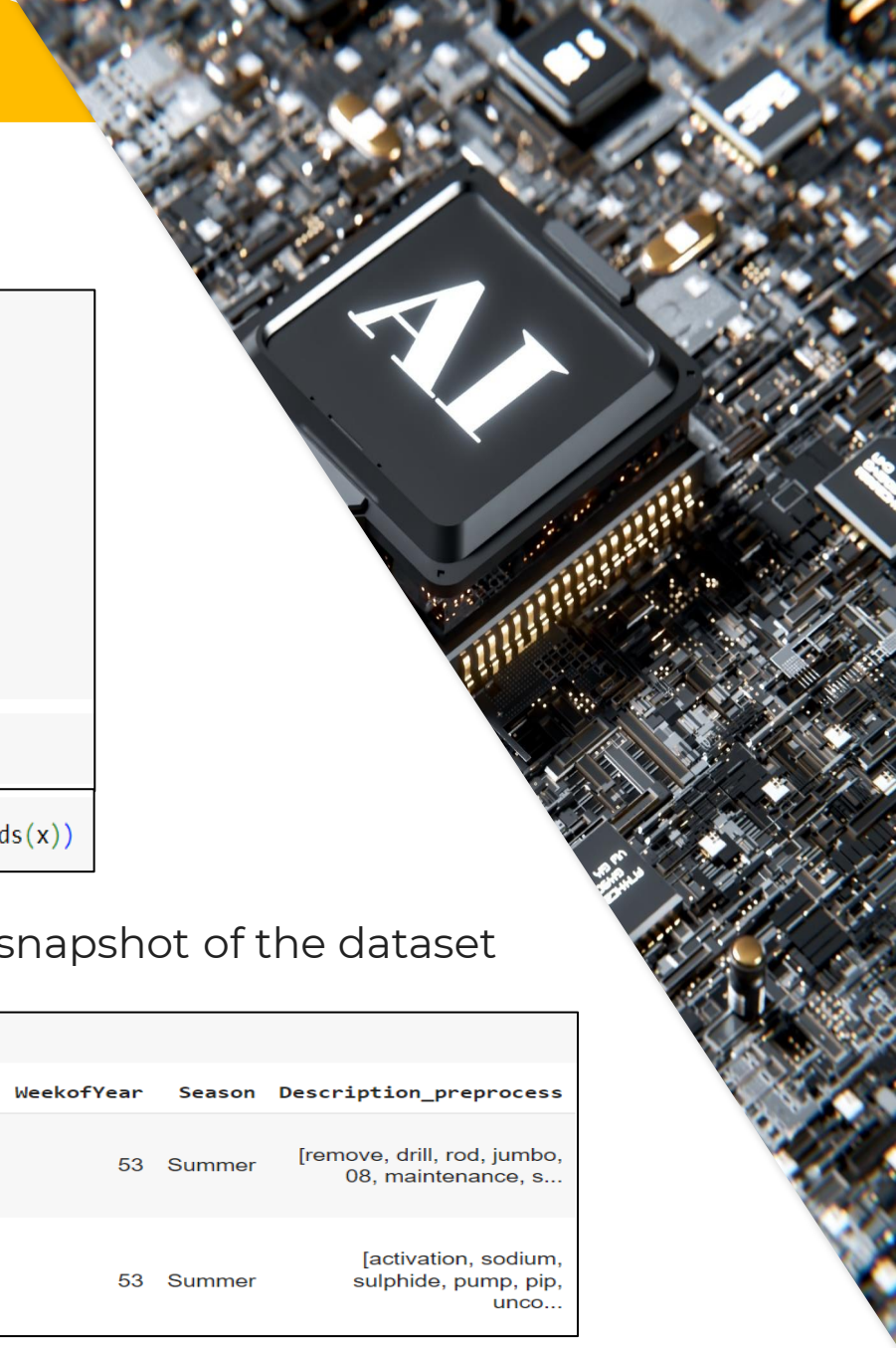
```
[ ]  chatbot_database['Description_preprocess'] = chatbot_database['Description_preprocess'].apply(lambda x :remove_whitespace(x))
```

6. Stemming

```
[ ]  stemmer = PorterStemmer()
     # stem words in the list of tokenised words
     def stem_words(text):
         word_tokens = word_tokenize(text)
         stems = [stemmer.stem(word) for word in word_tokens]
         return stems

     text = 'data science uses scientific methods algorithms and many types of processes'
     stem_words(text)
```



DATA PREPROCESSING



7. Lemmatization with tokenization

```
[ ]  # remove stopwords function
     def remove_stopwords(text):

          # Stop words
          stop_words = set(nltk.corpus.stopwords.words('english'))

          # Tokenization
          word_tokens = word_tokenize(text)

          # Final Text
          filtered_text = [word for word in word_tokens if word not in stop_words]

          # provide context i.e. part-of-speech - WordNetLemmatizer
          lemmatizer = WordNetLemmatizer()
          lemmas = [lemmatizer.lemmatize(word, pos ='v') for word in filtered_text]

          return lemmas
```

```
[ ]  example_text = "data science uses scientific methods algorithms and many types of processes"
     remove_stopwords(example_text)
```

```
[ ]  chatbot_database['Description_preprocess'] = chatbot_database['Description_preprocess'].apply(lambda x: remove_stopwords(x))
```

Result – After completing all the steps of data preprocessing, below is the snapshot of the dataset

```
chatbot_database.head()
```

ccident_Level	Potential_Accident_Level	Gender	Employee_type	Critical_Risk	Description	Year	Month	Day	Weekday	WeekofYear	Season	Description_preprocess
I	IV	Male	Third Party	Pressed	While removing the drill rod of the Jumbo 08 f...	2016	1	1	Friday	53	Summer	[remove, drill, rod, jumbo, 08, maintenance, s...
I	IV	Male	Employee	Pressurized Systems	During the activation of a sodium sulphide pum...	2016	1	2	Saturday	53	Summer	[activation, sodium, sulphide, pump, pip, unco...

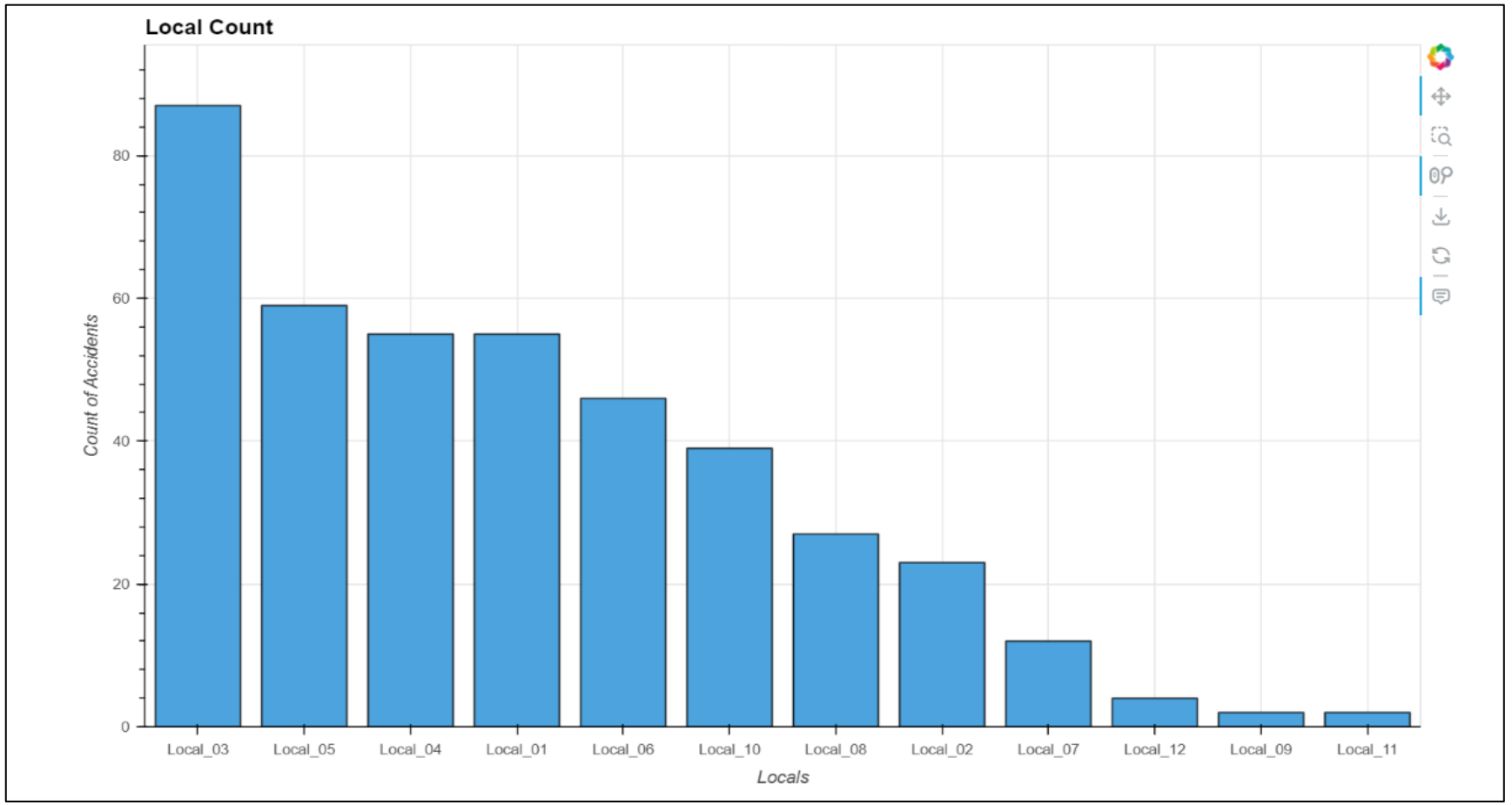
EXPLORATORY DATA ANALYSIS AND DATA PREPARATION

Following are the insights extracted by doing the EDA on the dataset :-

- Highest manufacturing plants are located in **Local_03 city** and lowest in **Local_09 city**.
- Percentage(%) of accidents occurred in respective countries: **59%** in Country_01, **31%** in Country_02 and **10%** in Country_03.
- Percentage(%) of manufacturing plants belongs to respective sectors: **56%** to Mining sector, **33%** to Metals sector and **11%** to Others sector.
- Metals and Mining industry sector plants are not available in **Country_03**. Distribution of industry sector differ significantly in each country.
- The number of accidents decreases as the Accident Level increases and increases as the Potential Accident Level increases.
- There are more men working in this industry as compared to women.
- **44%** Third party employees, **43%** own employees and **13%** Third party(Remote) employees working in this industry.
- Maximum number of accidents occurred in **2016** as compared to 2017.
- Number of accidents are high in **summer** and **autumn** and less in spring and winter
- Number of accidents increased during the middle of the week and declined after the middle of the week. Most of the critical risks are classified as Others.
- Most of the critical risks are classified as **Others**

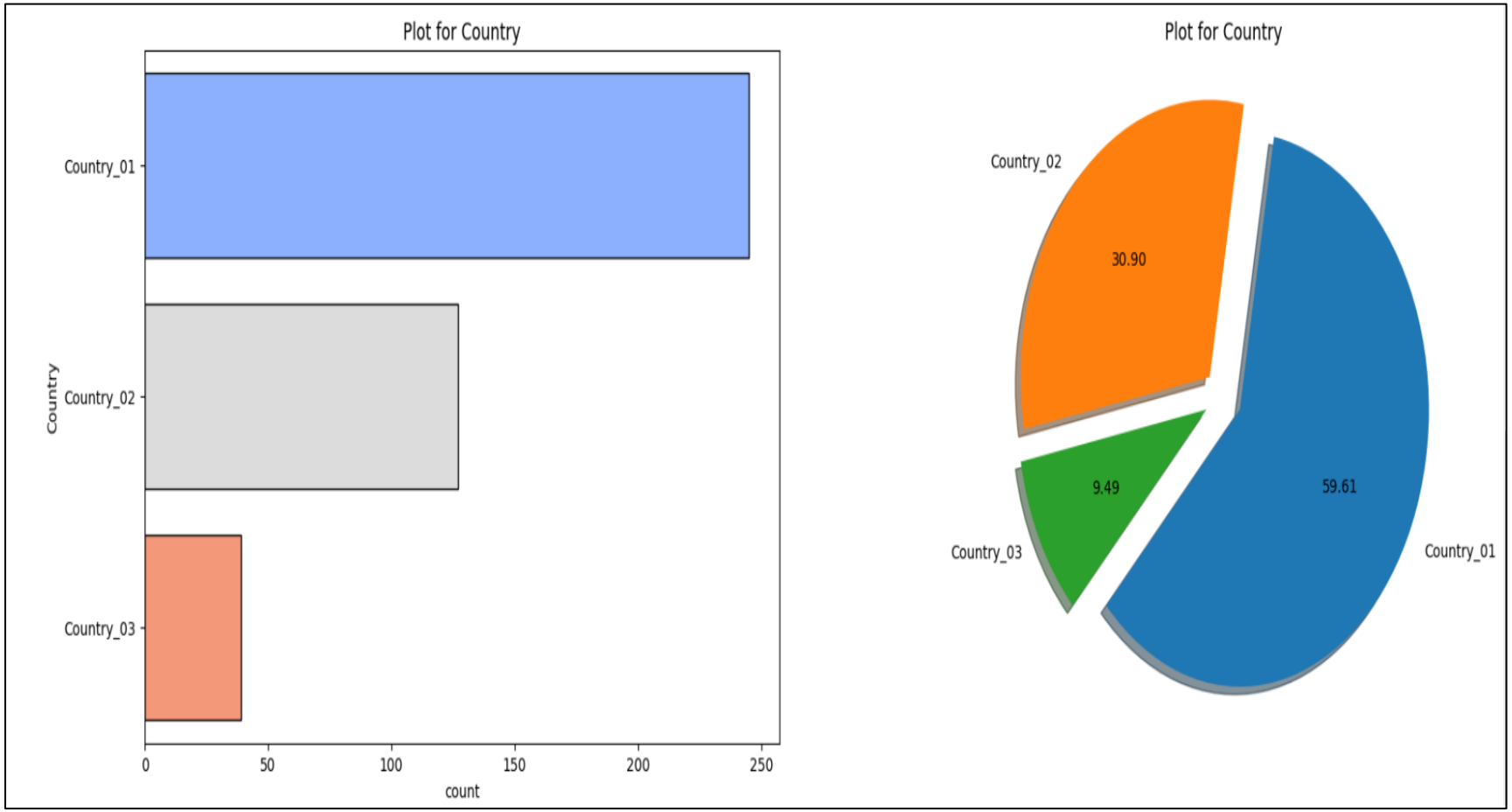
EXPLORATORY DATA ANALYSIS AND DATA PREPARATION

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



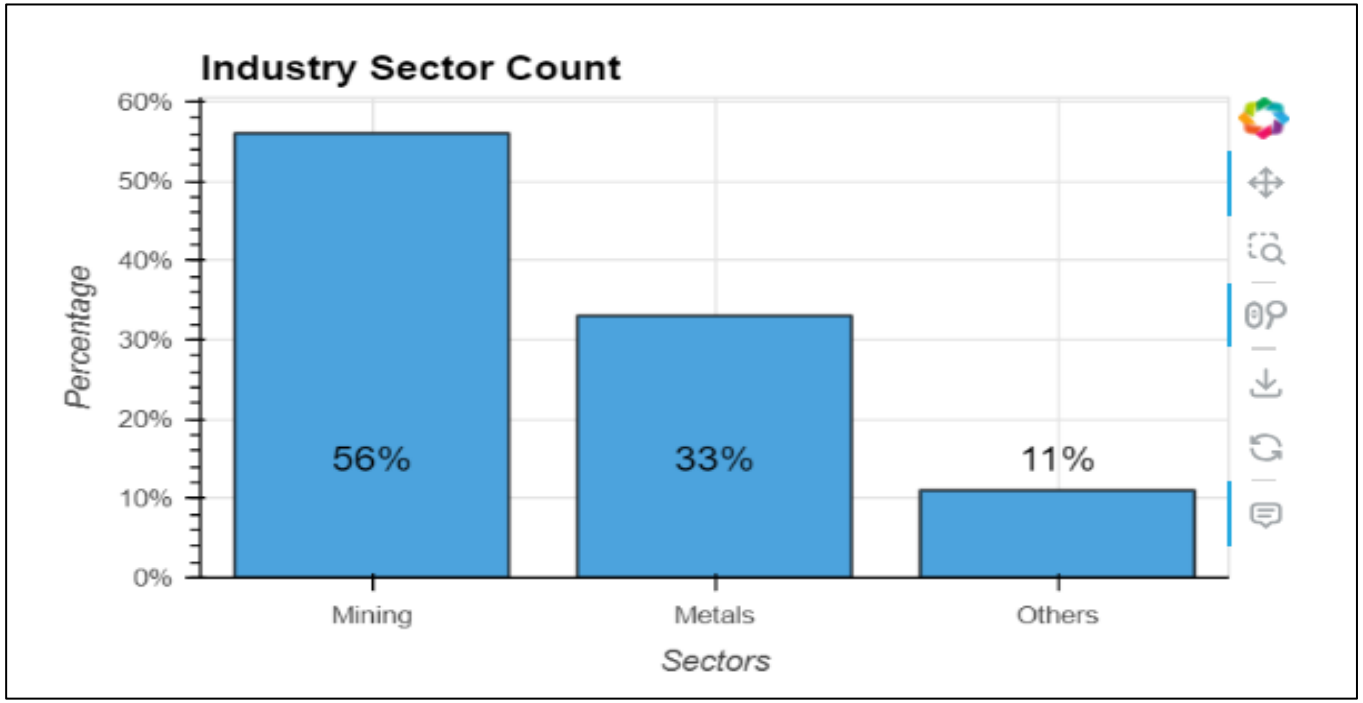
EXPLORATORY DATA ANALYSIS AND DATA PREPARATION

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



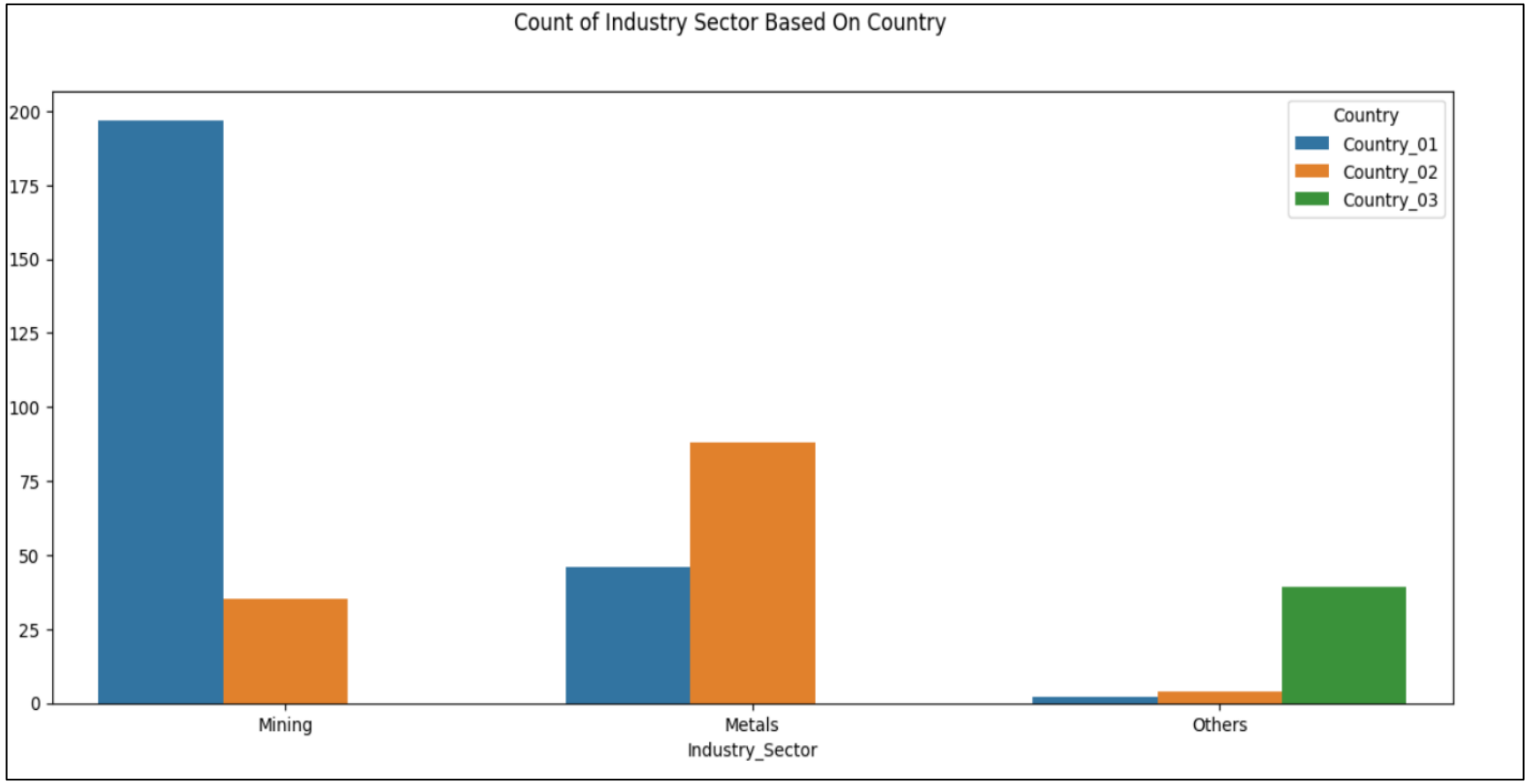
EXPLORATORY DATA ANALYSIS AND DATA PREPARATION

Percentage(%) of manufacturing plants belongs to respective sectors: 56% to Mining sector, 33% to Metals sector and 11% to Others sector :-



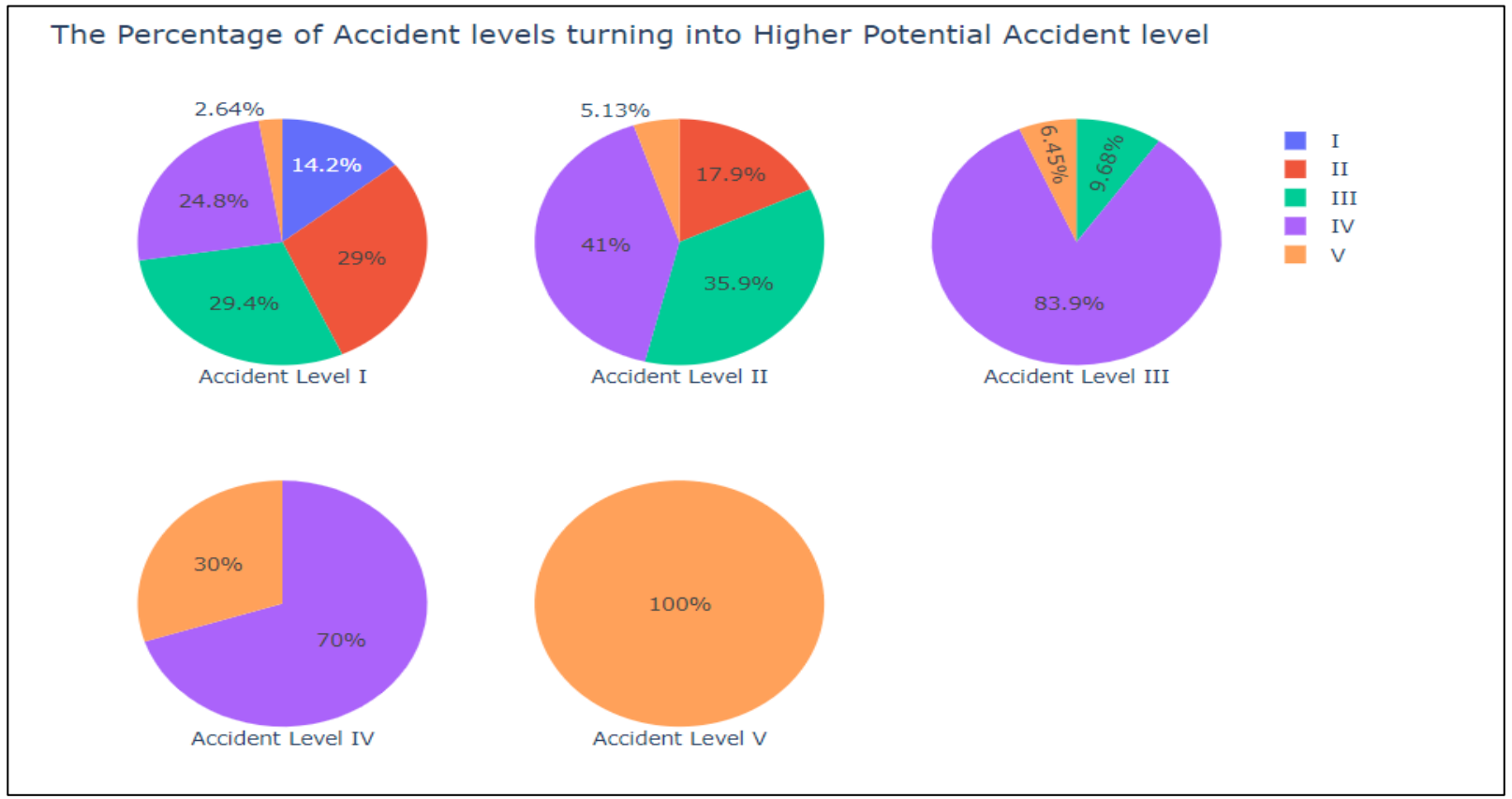
EXPLORATORY DATA ANALYSIS AND DATA PREPARATION

Metals and Mining industry sector plants are not available in Country_03. Distribution of industry sector differ significantly in each country :-



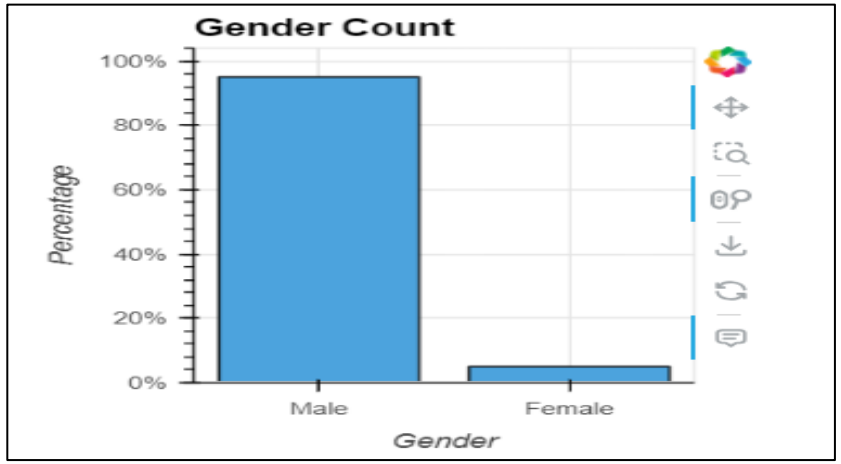
EXPLORATORY DATA ANALYSIS AND DATA PREPARATION

The number of accidents decreases as the Accident Level increases and increases as the Potential Accident Level increases :-

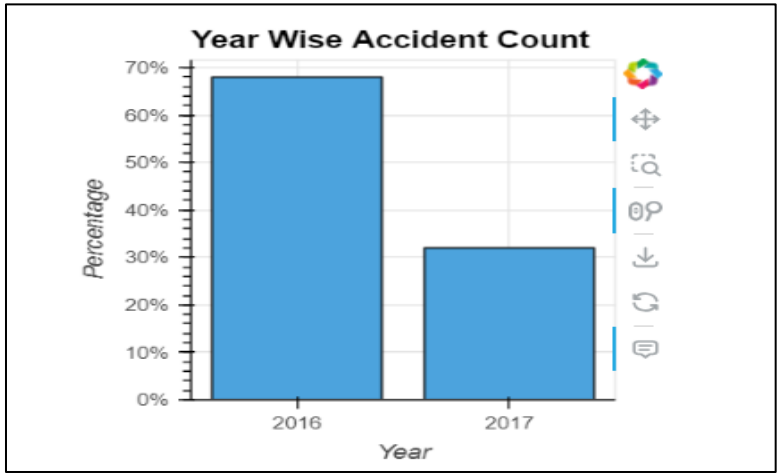


EXPLORATORY DATA ANALYSIS AND DATA PREPARATION

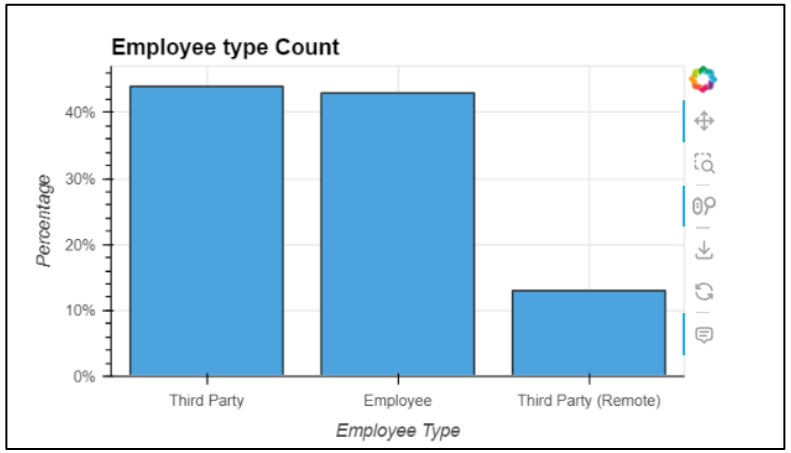
More men working as compared to women



Year 2016 has more accident than year 2017

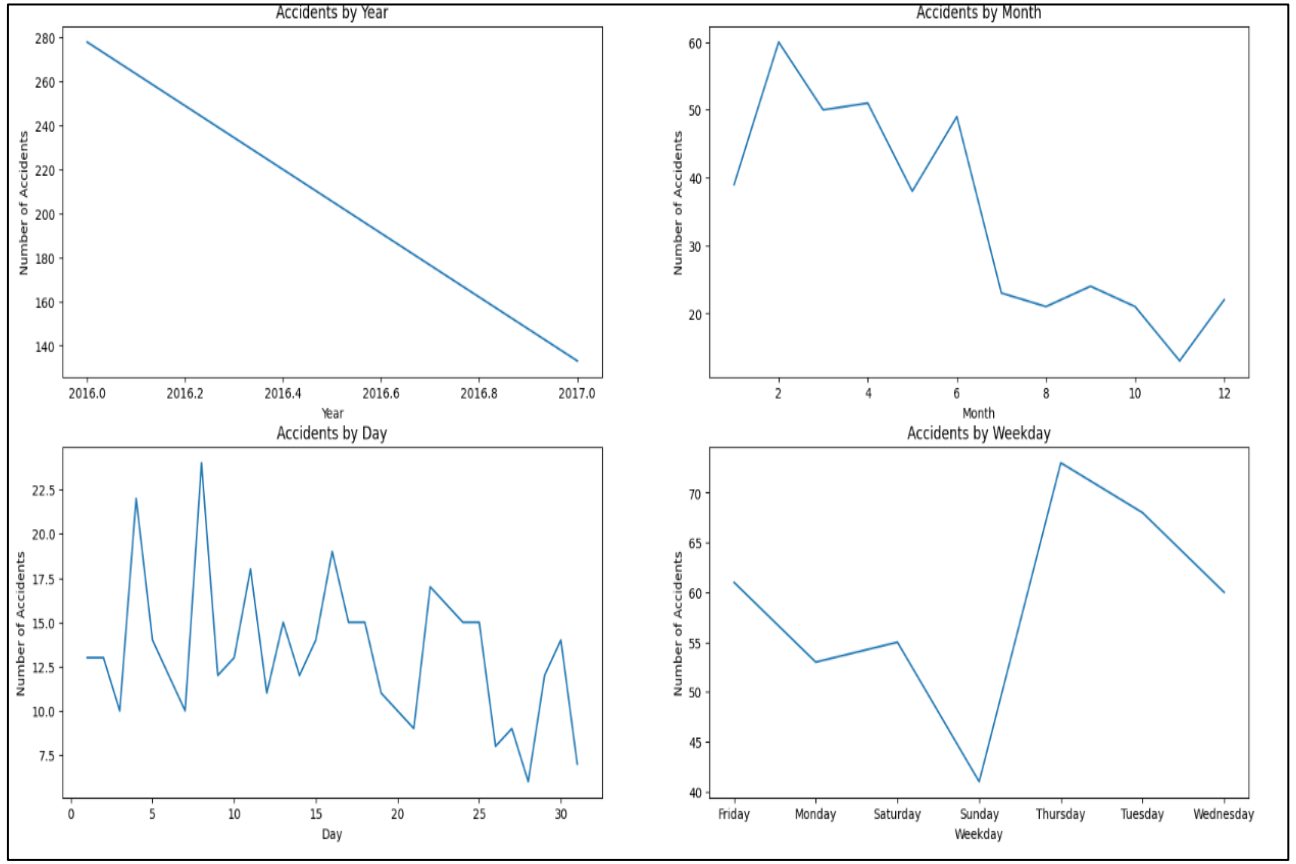


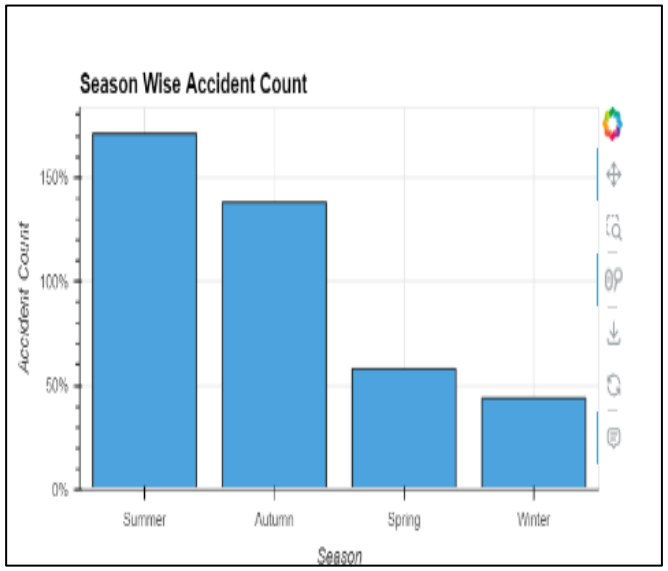
44% Third party employees, 43% own employees and 13% Third party(Remote) employees working in this industry:-



EXPLORATORY DATA ANALYSIS AND DATA PREPARATION

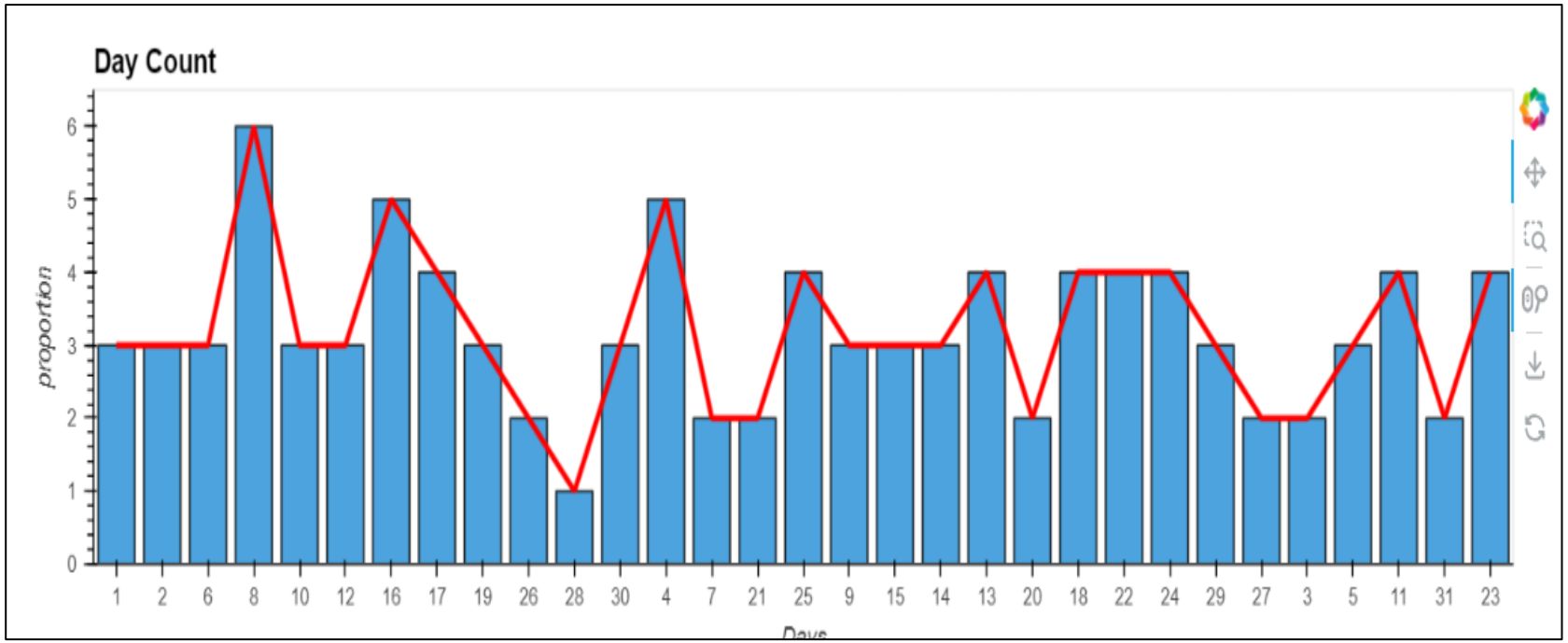
Number of accidents are high in summer and autumn and less in spring and winter





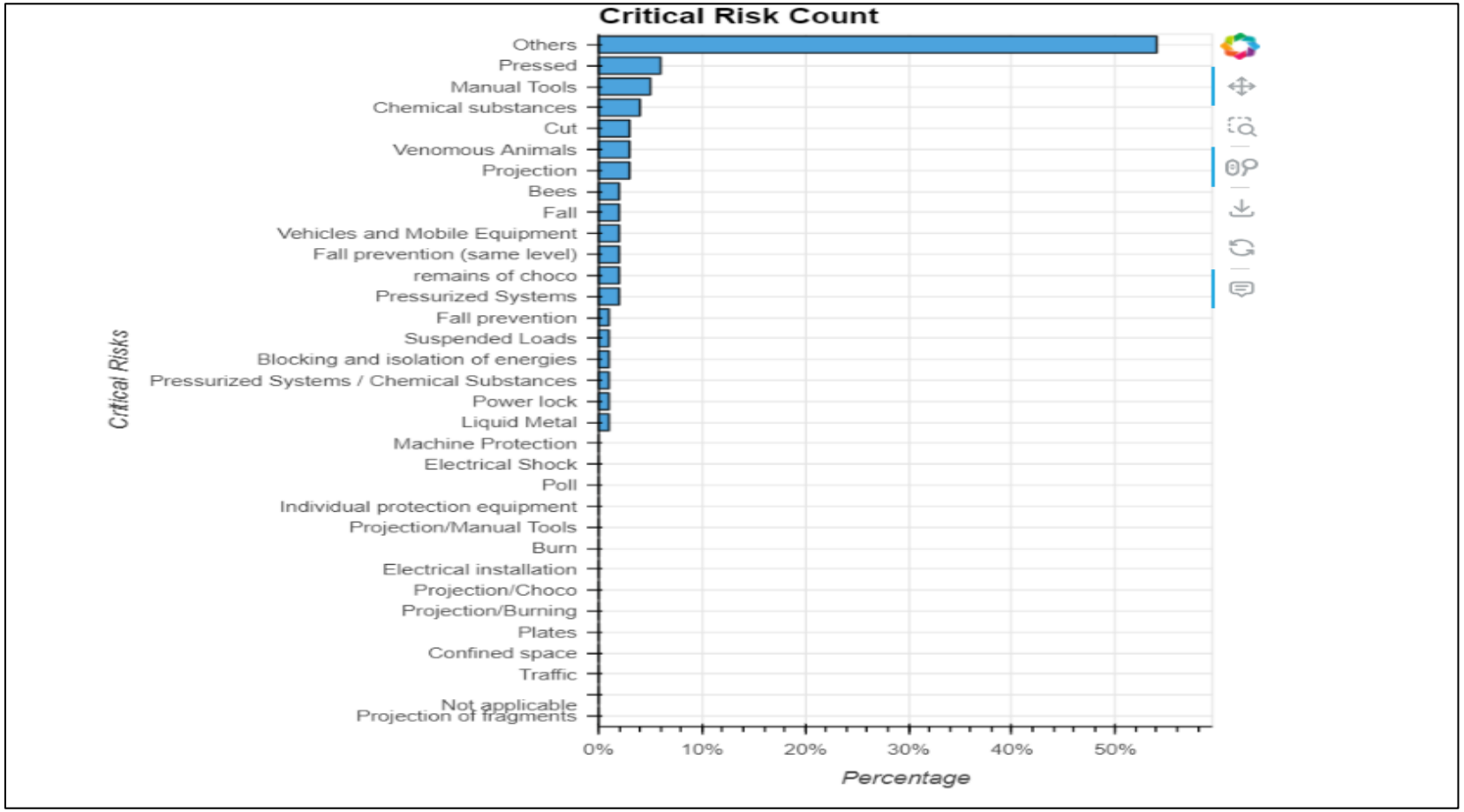
EXPLORATORY DATA ANALYSIS AND DATA PREPARATION

Number of accidents increased during the middle of the week and declined after the middle of the week. This indicates that workers are getting tried till middle of the week which cause more accidents



EXPLORATORY DATA ANALYSIS AND DATA PREPARATION

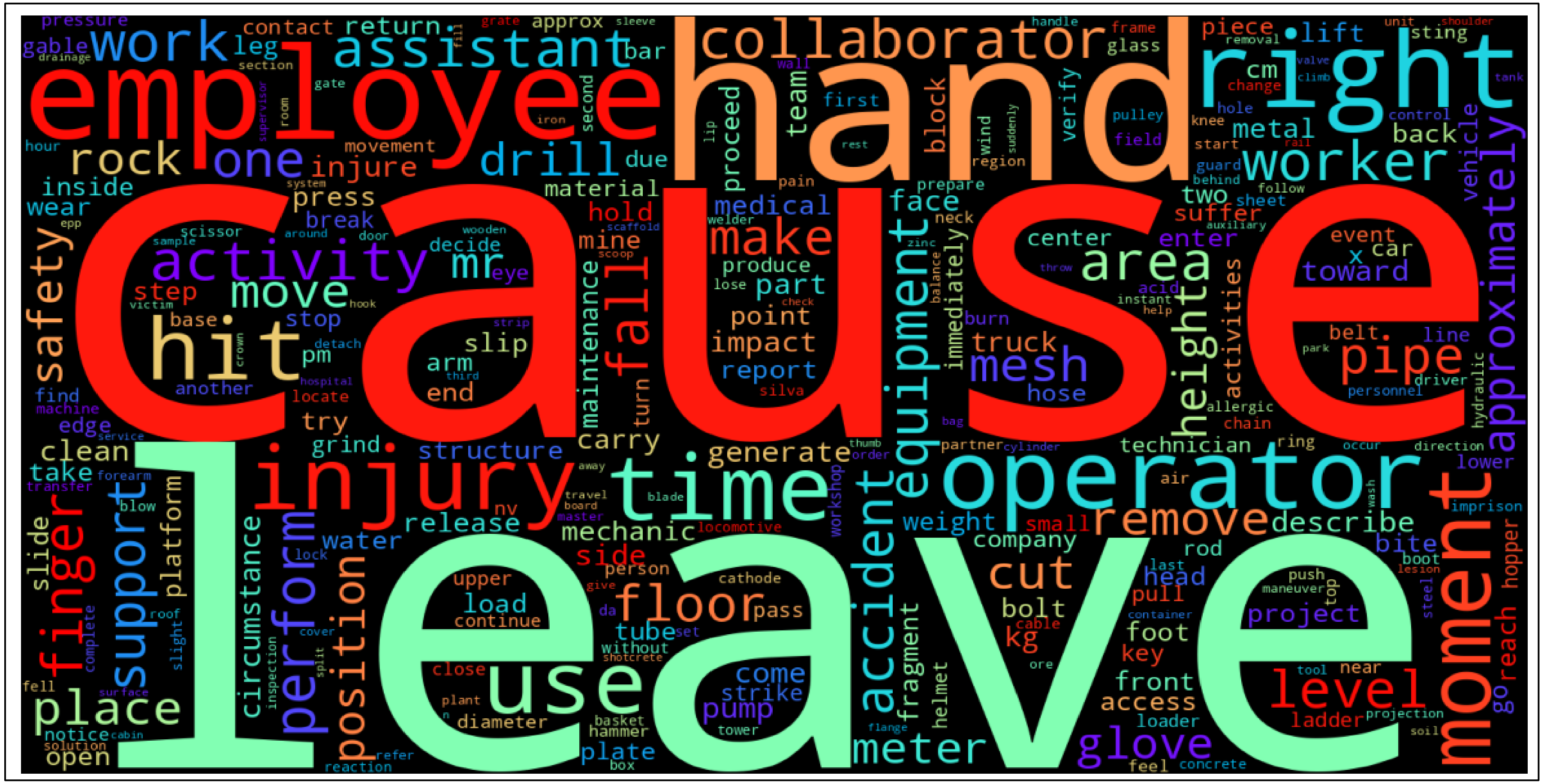
Most of the critical risks are classified as Others



EXPLORATORY DATA ANALYSIS AND DATA PREPARATION

Most frequent words in the accident descriptions

The most frequent words are cause, leave, hand, injury, and employee



DATA IMBALANCING TREATMENT

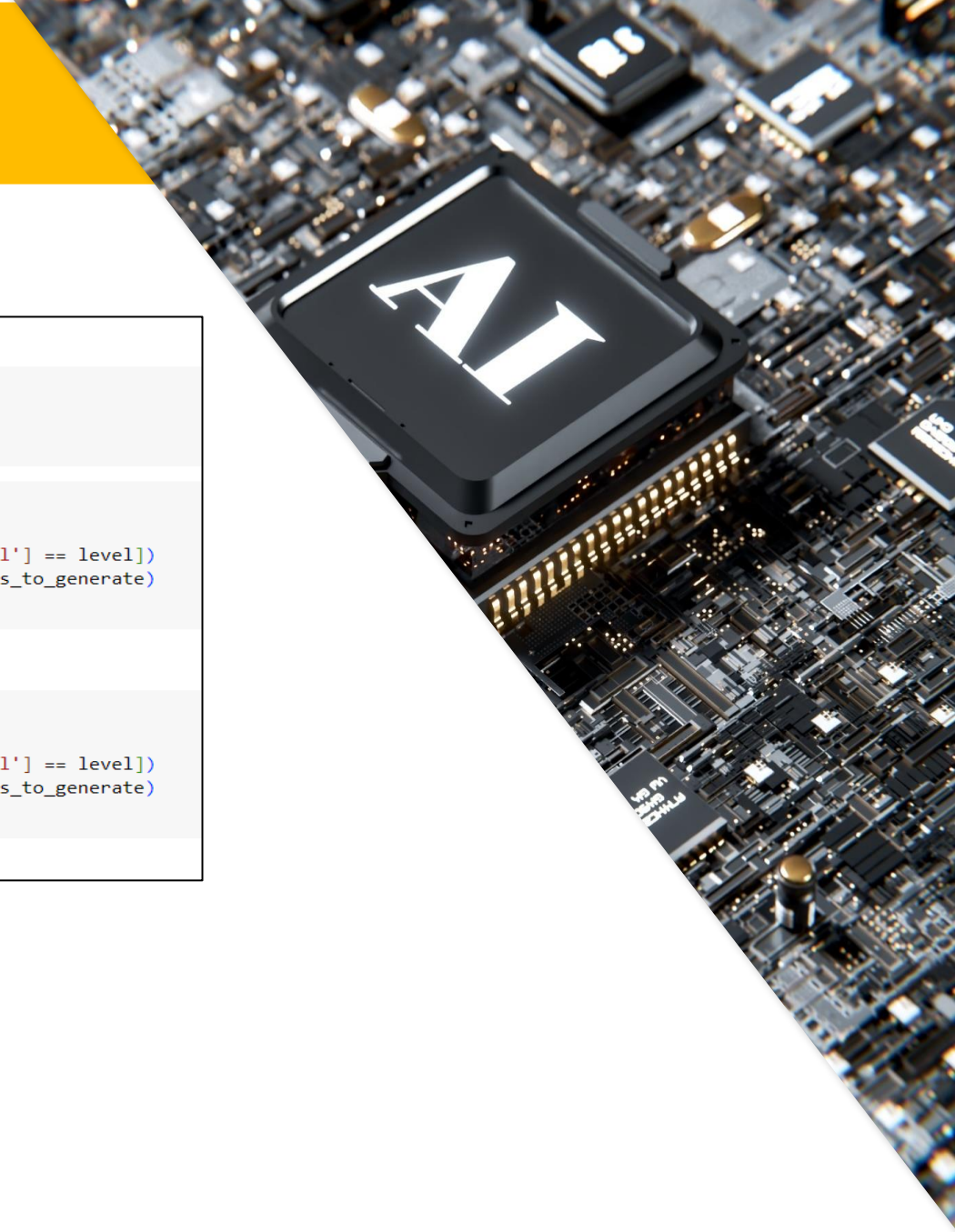
Below is the potential accident level data distribution which clearly indicates that we need to perform data imbalancing treatment :-

```
Potential_Accident_Level
IV     138
III    106
II      95
I       43
V       29
Name: count, dtype: int64
```

We have used GPT-2 to generate synthetic data for low count of accident level

```
[ ]  synthetic_data=pd.DataFrame()
     augmented_data=chatbot_database.copy()

[ ]  import pandas as pd
     import torch
     from transformers import GPT2Tokenizer, GPT2LMHeadModel
     from tqdm import tqdm

[ ]  device=torch.device('cuda' if torch.cuda.is_available() else 'cpu')

[ ]  import os
     os.environ["OPENAI_API_KEY"] = "OPENAI_API_KEY"

     model_name='gpt2'
     model=GPT2LMHeadModel.from_pretrained(model_name).to(device)
     tokenizer=GPT2Tokenizer.from_pretrained(model_name)
```

DATA IMBALANCING TREATMENT



We have generated synthetic data for all accident level

Generate Synthetic samples for Potential accident level

```
[ ] # @title Generate Synthetic samples for Potential accident level
    synthetic_data=pd.DataFrame()
    augmented_data=chatbot_database.copy()
```

```
[ ] from tqdm import tqdm
    for level in tqdm(minority_potential_levels, desc="Generating remaining synthetic samples"):
        num_samples_to_generate = majority_class_count_pal-len(augmented_data[augmented_data['Potential_Accident_Level'] == level])
        synthetic_samples = generate_synthetic_samples(chatbot_database, 'Potential_Accident_Level', level,num_samples_to_generate)
        augmented_data = pd.concat([augmented_data, synthetic_samples], ignore_index=True)
```

```
Generating remaining synthetic samples: 100%|██████████| 3/3 [13:28<00:00, 269.62s/it]
```

```
[ ] from tqdm import tqdm
    for level in tqdm(minority_potential_levels, desc="Generating remaining synthetic samples"):
        num_samples_to_generate = majority_class_count_pal-len(augmented_data[augmented_data['Potential_Accident_Level'] == level])
        synthetic_samples = generate_synthetic_samples(chatbot_database, 'Potential_Accident_Level', level,num_samples_to_generate)
        augmented_data = pd.concat([augmented_data, synthetic_samples], ignore_index=True)
```

```
Generating remaining synthetic samples: 100%|██████████| 3/3 [07:05<00:00, 141.75s/it]
```

Below is the updated count for each potential accident level data

```
Potential_Accident_Level
IV      138
III     138
I       138
II      138
V       138
Name: count, dtype: int64
```

MODEL BUILDING AND EVALUATION

We have taken Potential_Accident_Level as target column and build 10 machine learning models

```
def ml_models(X_train, y_train, X_test, y_test):
    # Dictionary with different ML models
    models = {
        'LogReg': LogisticRegression(),
        'Naive Bayes': GaussianNB(),
        'KNN': KNeighborsClassifier(),
        'SVM': SVC(),
        'Decision Tree': DecisionTreeClassifier(criterion='entropy', max_depth=6, random_state=100, min_samples_leaf=5),
        'RandomForest': RandomForestClassifier(n_estimators=100, max_depth=7),
        'Bagging': BaggingClassifier(n_estimators=50, max_samples=0.7),
        'AdaBoost': AdaBoostClassifier(n_estimators=50),
        'Gradient Boost': GradientBoostingClassifier(n_estimators=50, learning_rate=0.05),
        'XGBoost': XGBClassifier(enable_categorical=True) # Enable categorical data handling for XGBoost
    }

    names = []
    train_accuracy = []
    test_accuracy = []
    F1_score = []
    Precision = []
    Recall = []

    for name, model in models.items(): # Looping through each model
        clf = model.fit(X_train, y_train) # Fit the models one by one
        tr_accuracy = clf.score(X_train, y_train)
        te_accuracy = clf.score(X_test, y_test)
        y_pred = clf.predict(X_test)
        F1_sre = f1_score(y_test, y_pred, average='weighted')
        precision = precision_score(y_test, y_pred, average='weighted')
        recall = recall_score(y_test, y_pred, average='weighted')

        print(f"Model: {name}")
        print(classification_report(y_test, y_pred))
        cm_display = ConfusionMatrixDisplay.from_estimator(model, X_test, y_test)
        cm_display.plot()

        names.append(name)
        train_accuracy.append(tr_accuracy)
        test_accuracy.append(te_accuracy)
        F1_score.append(F1_sre)
        Precision.append(precision)
        Recall.append(recall)

    combined_model_result = pd.DataFrame({
        'Model': names,
        'Train_accuracy': train_accuracy,
        'Test_accuracy': test_accuracy,
        'F1_score': F1_score,
        'Precision': Precision,
        'Recall': Recall
    })

    return combined_model_result # Returns the dataframe
```

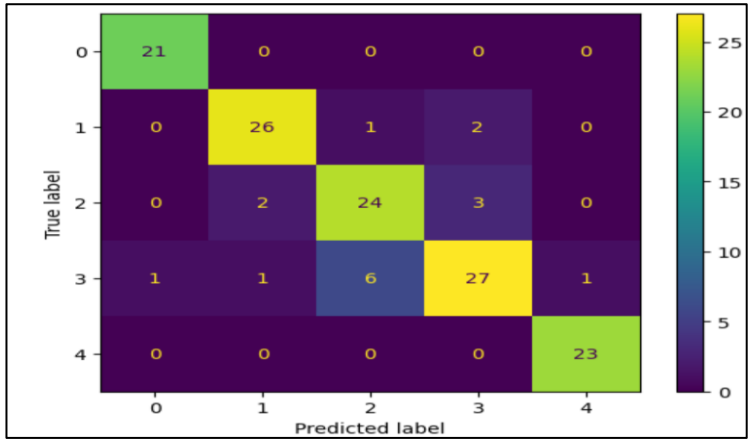
MODEL BUILDING AND EVALUATION

Below are the result of the 10 machine learning models:-

Logistic Regression

Model: LogReg

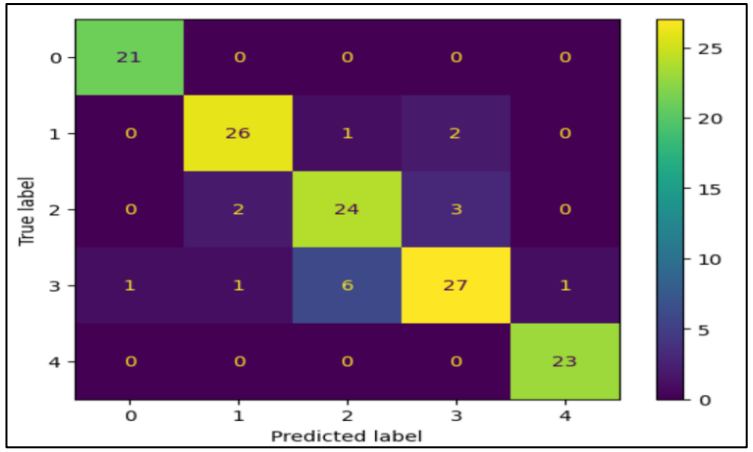
	precision	recall	f1-score	support
0	0.95	1.00	0.98	21
1	0.90	0.90	0.90	29
2	0.77	0.83	0.80	29
3	0.84	0.75	0.79	36
4	0.96	1.00	0.98	23
accuracy			0.88	138
macro avg	0.89	0.89	0.89	138
weighted avg	0.88	0.88	0.88	138

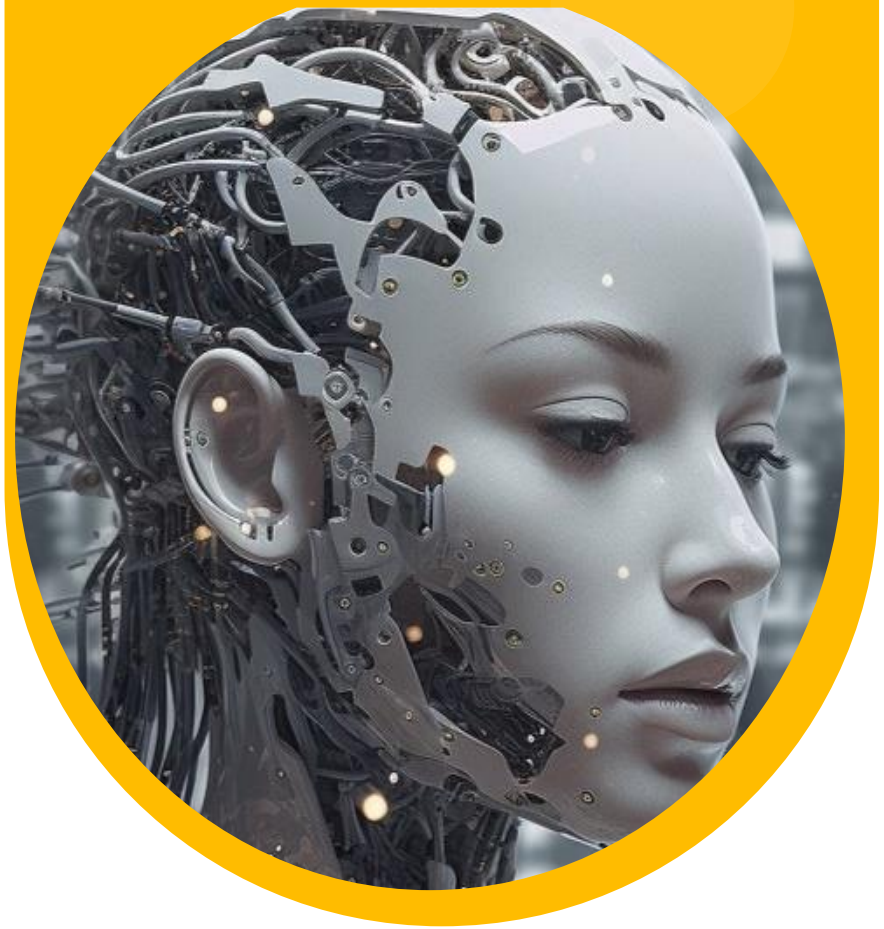


Naive Bayes

Model: Naive Bayes

	precision	recall	f1-score	support
0	0.95	1.00	0.98	21
1	0.87	0.93	0.90	29
2	0.91	0.72	0.81	29
3	0.79	0.83	0.81	36
4	0.96	1.00	0.98	23
accuracy			0.88	138
macro avg	0.90	0.90	0.89	138
weighted avg	0.89	0.88	0.88	138





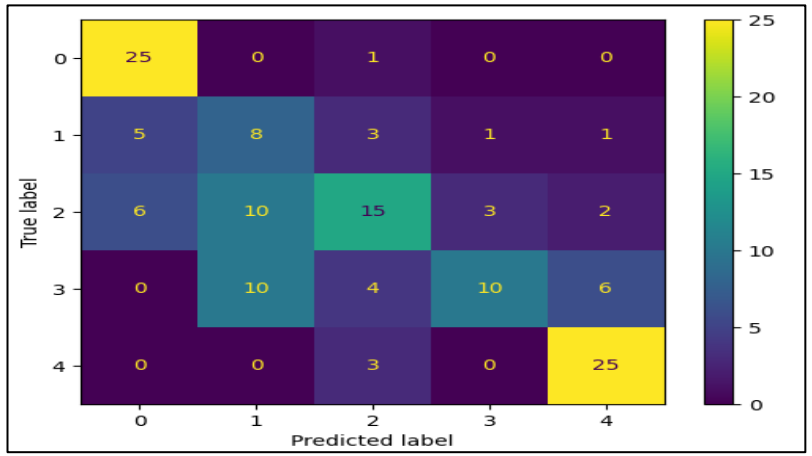
MODEL BUILDING AND EVALUATION

Below are the result of the 10 machine learning models:-

KNN

Model: KNN

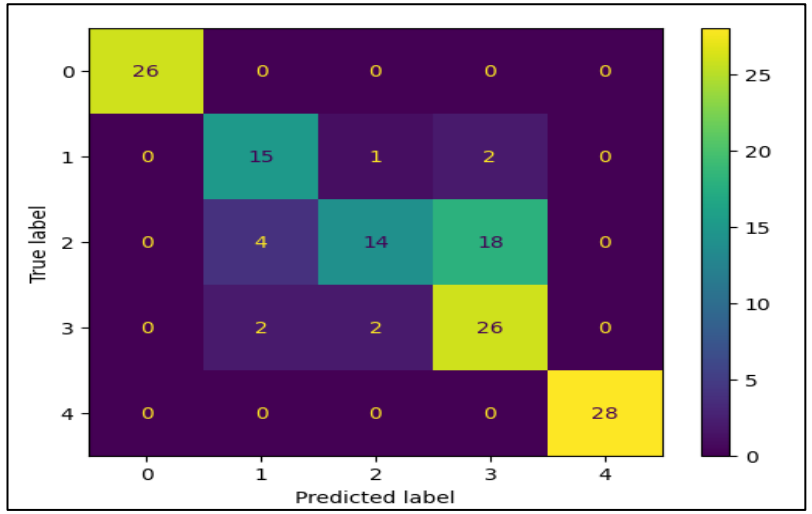
	precision	recall	f1-score	support
0	0.69	0.96	0.81	26
1	0.29	0.44	0.35	18
2	0.58	0.42	0.48	36
3	0.71	0.33	0.45	30
4	0.74	0.89	0.81	28
accuracy			0.60	138
macro avg	0.60	0.61	0.58	138
weighted avg	0.62	0.60	0.59	138

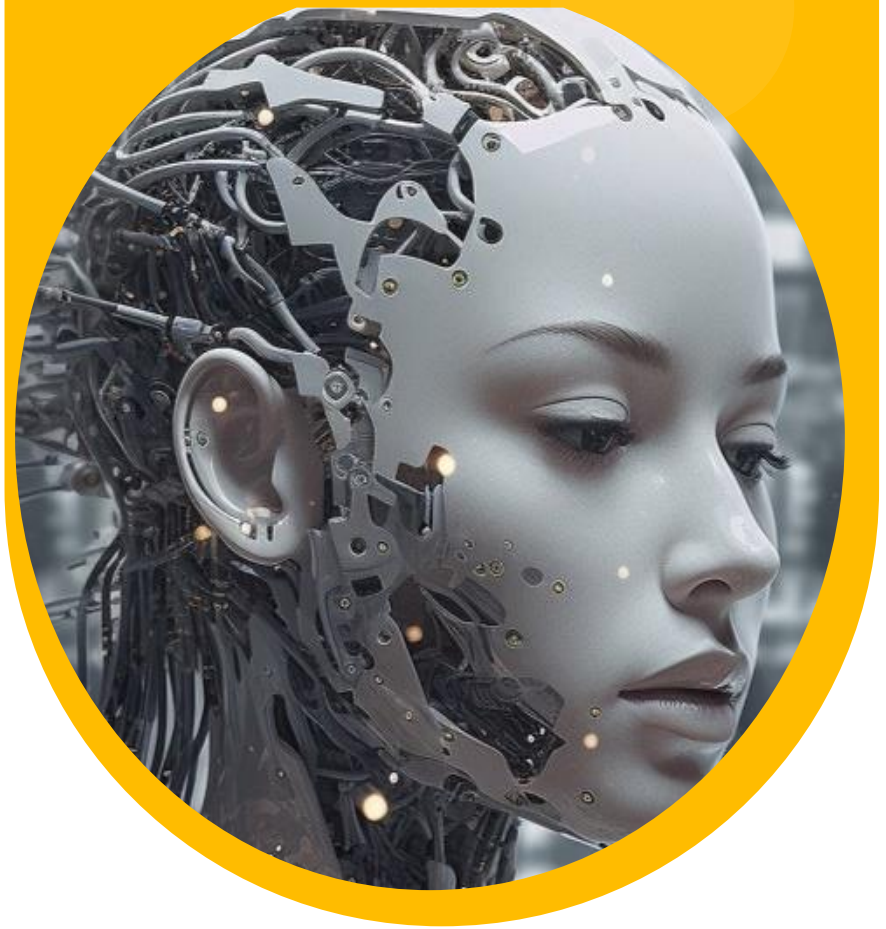


SVM

Model: SVM

	precision	recall	f1-score	support
0	1.00	1.00	1.00	26
1	0.71	0.83	0.77	18
2	0.82	0.39	0.53	36
3	0.57	0.87	0.68	30
4	1.00	1.00	1.00	28
accuracy			0.79	138
macro avg	0.82	0.82	0.80	138
weighted avg	0.82	0.79	0.78	138





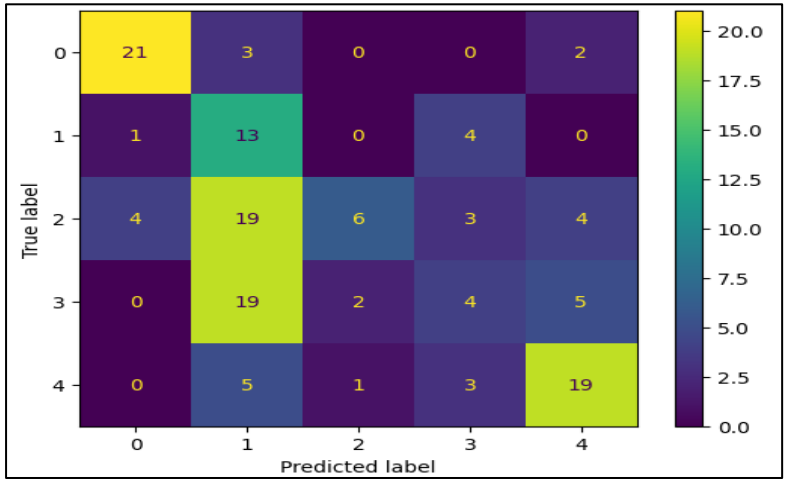
MODEL BUILDING AND EVALUATION

Below are the result of the 10 machine learning models:-

Decision Tree

Model: Decision Tree

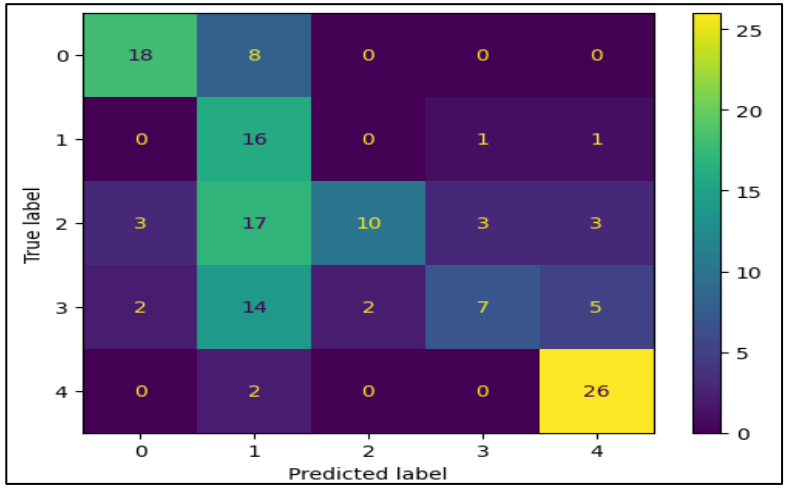
	precision	recall	f1-score	support
0	0.81	0.81	0.81	26
1	0.22	0.72	0.34	18
2	0.67	0.17	0.27	36
3	0.29	0.13	0.18	30
4	0.63	0.68	0.66	28
accuracy			0.46	138
macro avg	0.52	0.50	0.45	138
weighted avg	0.55	0.46	0.44	138

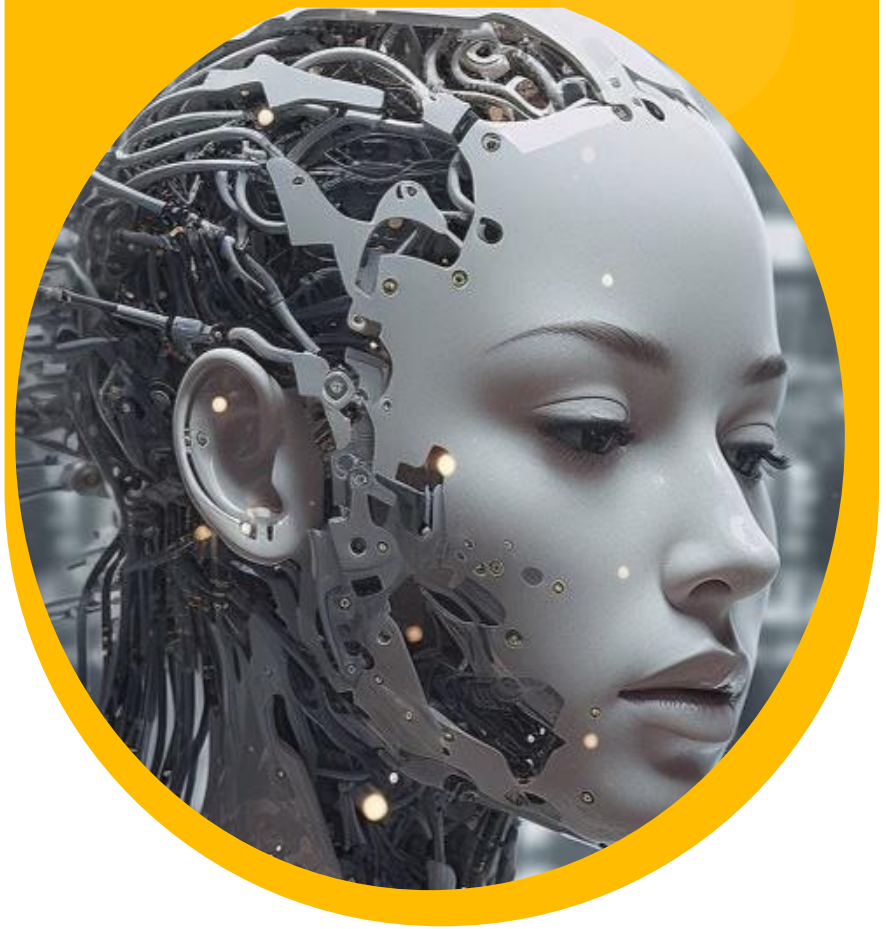


Random Forest

Model: RandomForest

	precision	recall	f1-score	support
0	0.78	0.69	0.73	26
1	0.28	0.89	0.43	18
2	0.83	0.28	0.42	36
3	0.64	0.23	0.34	30
4	0.74	0.93	0.83	28
accuracy			0.56	138
macro avg	0.66	0.60	0.55	138
weighted avg	0.69	0.56	0.54	138





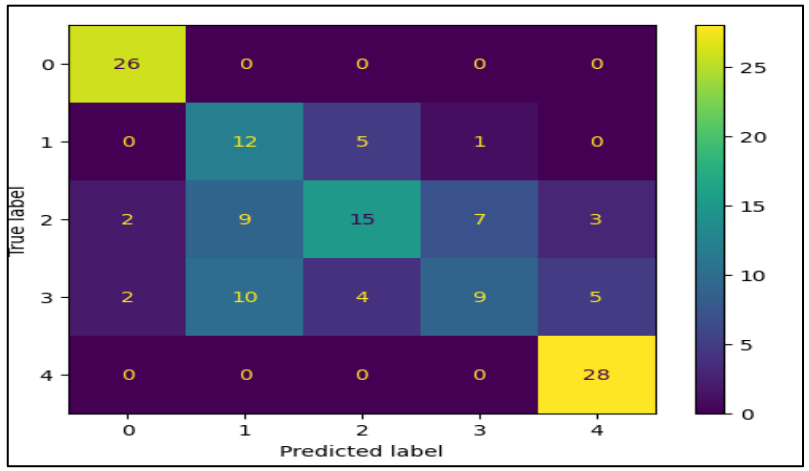
MODEL BUILDING AND EVALUATION

Below are the result of the 10 machine learning models:-

Bagging

Model: Bagging

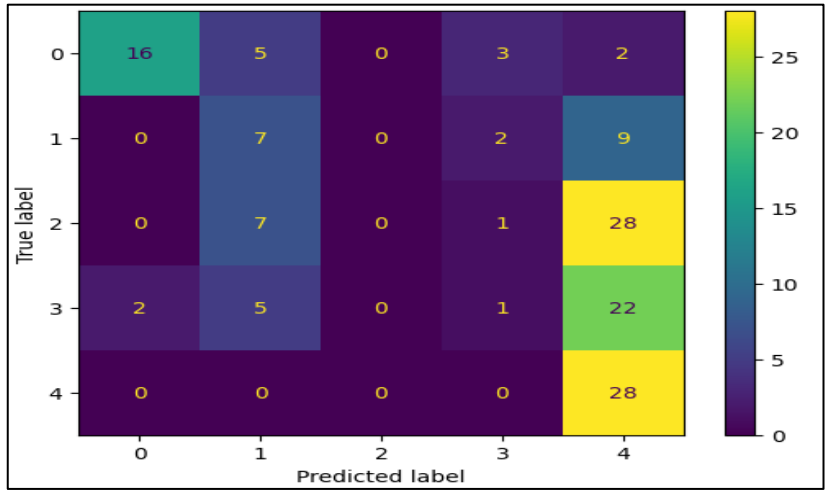
	precision	recall	f1-score	support
0	0.87	1.00	0.93	26
1	0.39	0.67	0.49	18
2	0.62	0.42	0.50	36
3	0.53	0.30	0.38	30
4	0.78	1.00	0.88	28
accuracy			0.65	138
macro avg	0.64	0.68	0.64	138
weighted avg	0.65	0.65	0.63	138

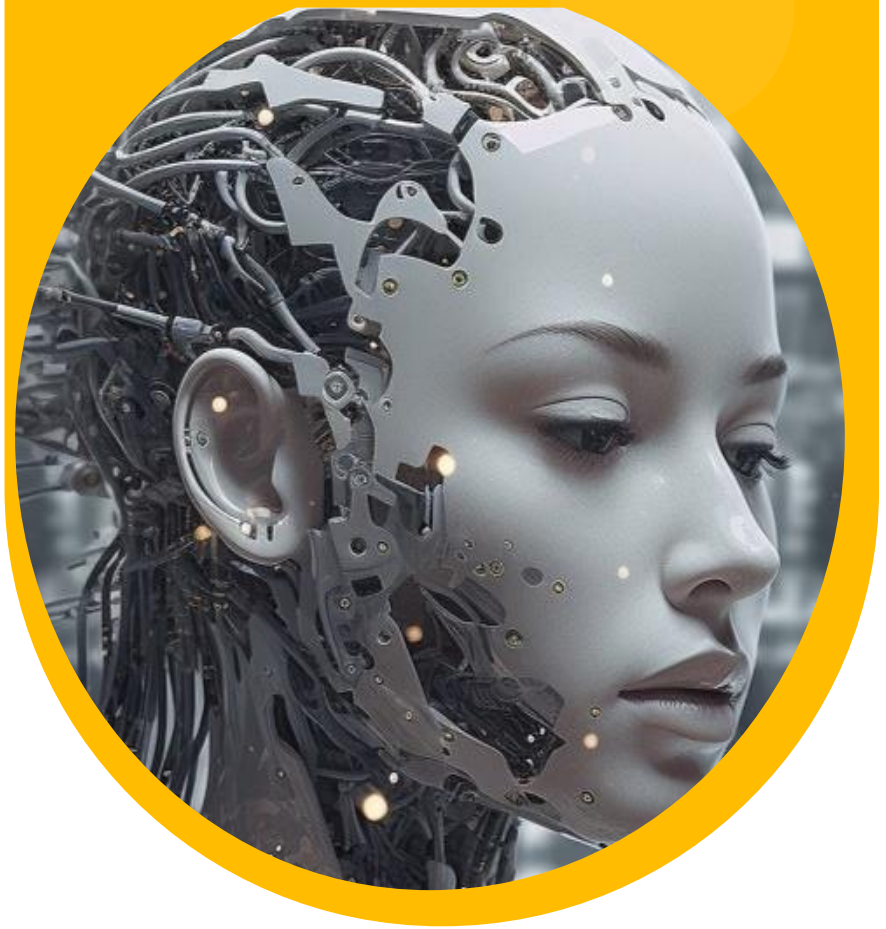


AdaBoost

Model: AdaBoost

	precision	recall	f1-score	support
0	0.89	0.62	0.73	26
1	0.29	0.39	0.33	18
2	0.00	0.00	0.00	36
3	0.14	0.03	0.05	30
4	0.31	1.00	0.48	28
accuracy			0.38	138
macro avg	0.33	0.41	0.32	138
weighted avg	0.30	0.38	0.29	138





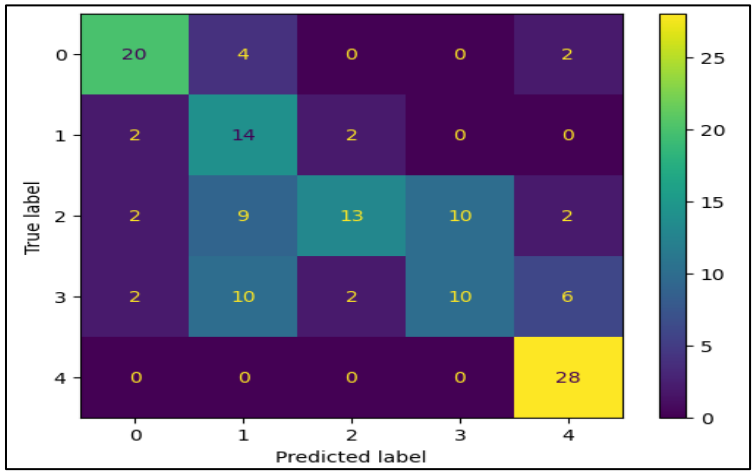
MODEL BUILDING AND EVALUATION

Below are the result of the 10 machine learning models:-

Gradient Boost

Model: Gradient Boost

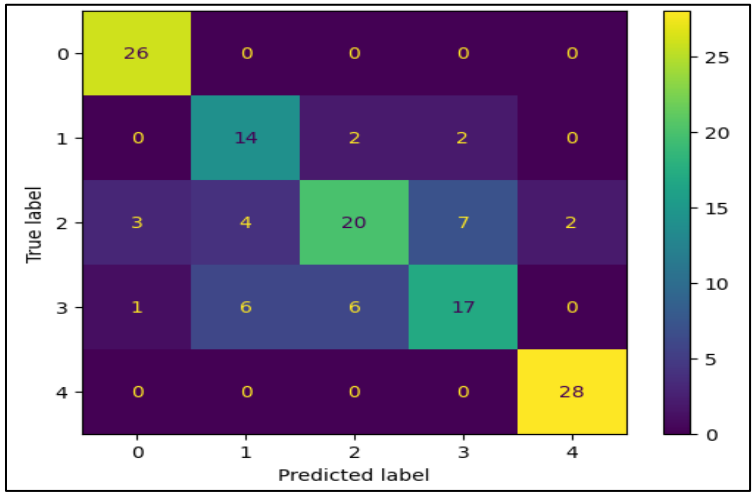
	precision	recall	f1-score	support
0	0.77	0.77	0.77	26
1	0.38	0.78	0.51	18
2	0.76	0.36	0.49	36
3	0.50	0.33	0.40	30
4	0.74	1.00	0.85	28
accuracy			0.62	138
macro avg	0.63	0.65	0.60	138
weighted avg	0.65	0.62	0.60	138

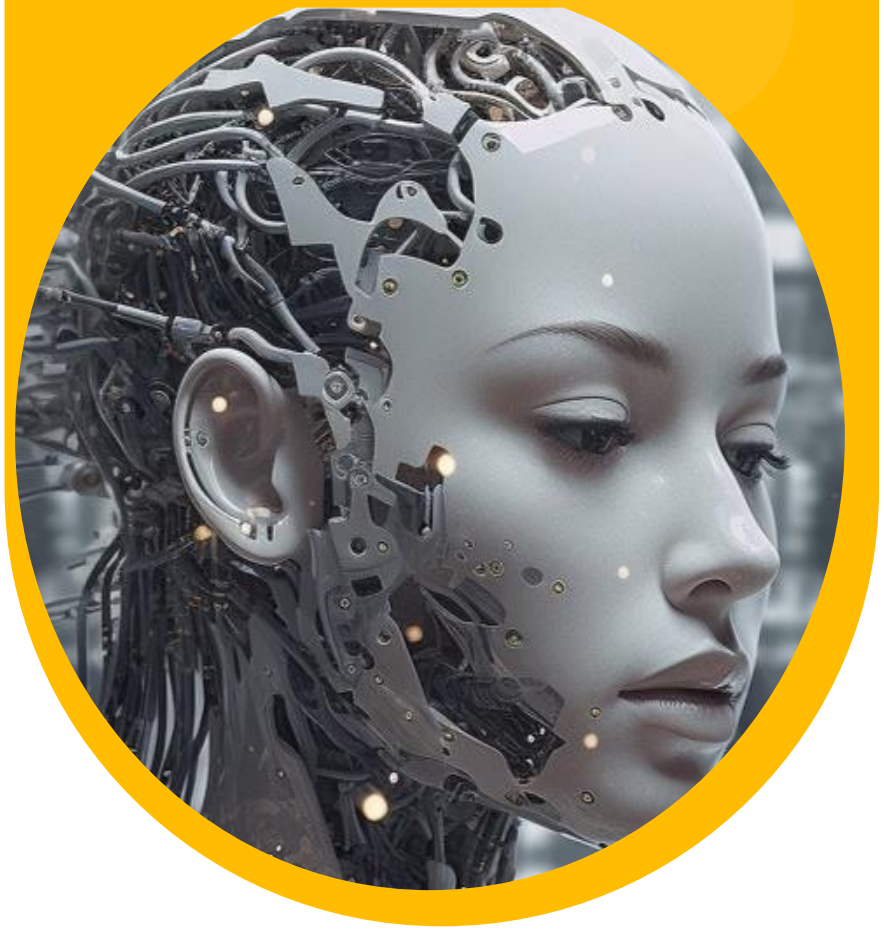


XGBoost

Model: XGBoost

	precision	recall	f1-score	support
0	0.87	1.00	0.93	26
1	0.58	0.78	0.67	18
2	0.71	0.56	0.63	36
3	0.65	0.57	0.61	30
4	0.93	1.00	0.97	28
accuracy			0.76	138
macro avg	0.75	0.78	0.76	138
weighted avg	0.76	0.76	0.75	138





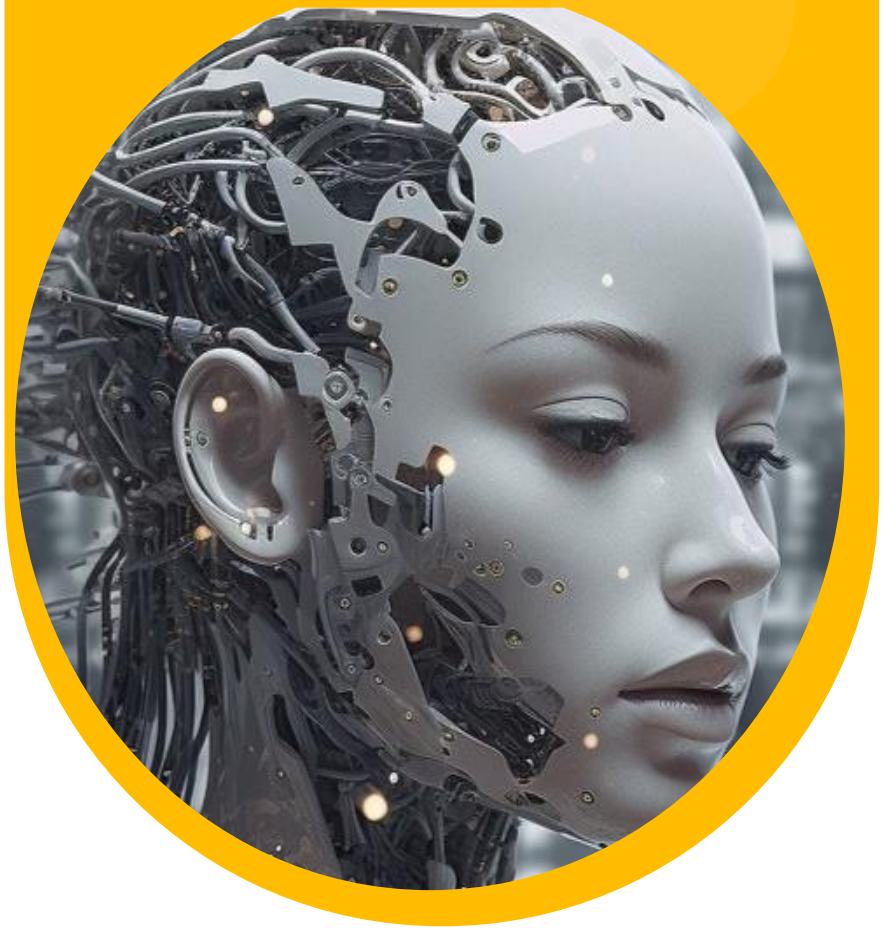
MODEL BUILDING AND EVALUATION

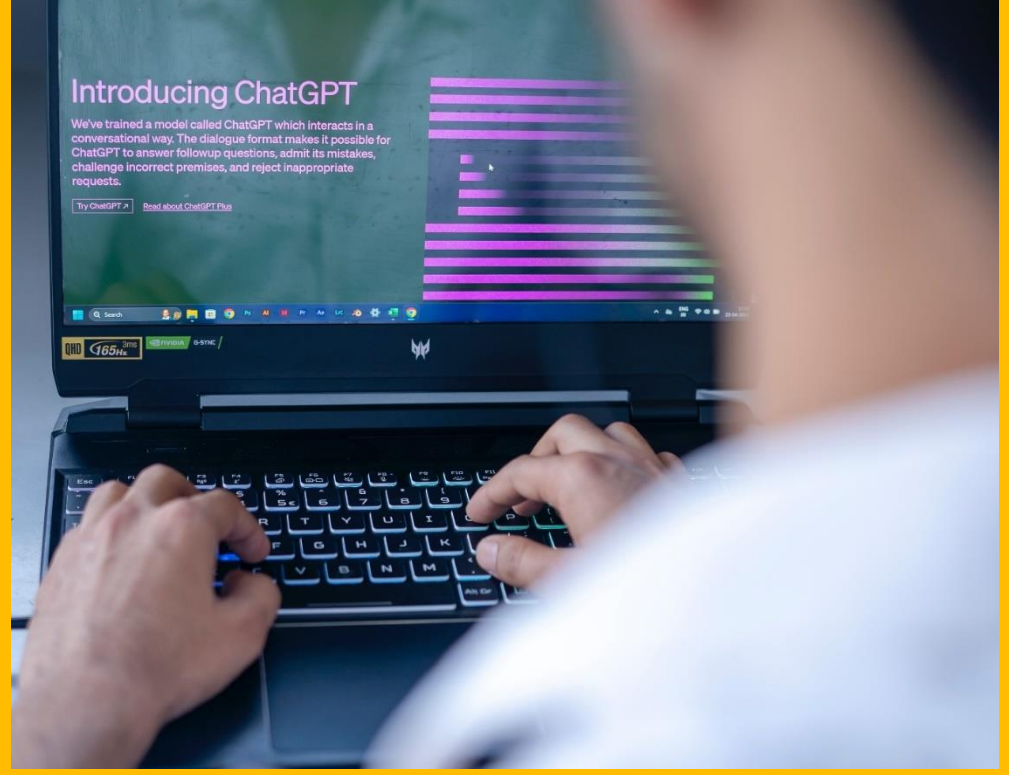
Final result of all models :-

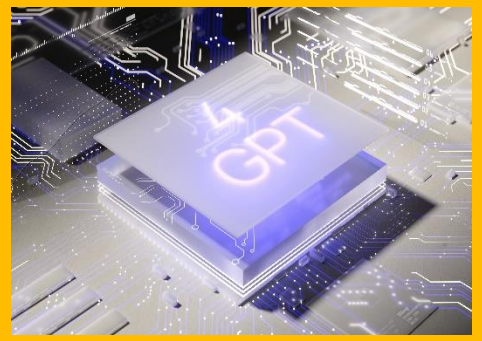
```
In [ ]: combined_model_result
```

Out[ ]:

	Model	Train_accuracy	Test_accuracy	F1_score	Precision	Recall
0	LogReg	0.992754	0.876812	0.875438	0.876186	0.876812
1	Naive Bayes	1.000000	0.884058	0.882134	0.885830	0.884058
2	KNN	0.750000	0.572464	0.543475	0.567133	0.572464
3	SVM	1.000000	0.942029	0.942376	0.945282	0.942029
4	Decision Tree	0.585145	0.449275	0.412969	0.629424	0.449275
5	RandomForest	0.922101	0.775362	0.770077	0.804769	0.775362
6	Bagging	1.000000	0.862319	0.859724	0.877048	0.862319
7	AdaBoost	0.364130	0.326087	0.274784	0.360186	0.326087
8	Gradient Boost	0.958333	0.782609	0.780698	0.792245	0.782609
9	XGBoost	1.000000	0.891304	0.888376	0.891074	0.891304





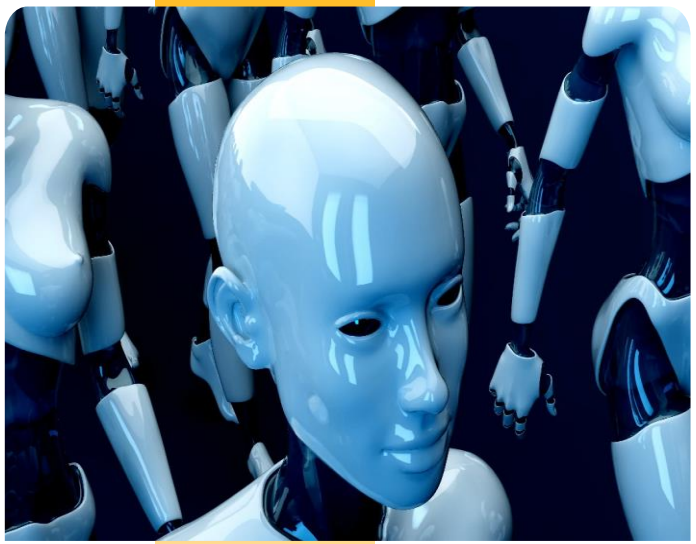


Milestone-1 Conclusion

Below are the steps we have taken in the process of making a ML/DL based chatbot for the professionals to highlight the safety risk :-

- Renamed the columns to understand the features of the dataset
- Removed duplicate values from the dataset
- Checked missing values and outliers
- Executed major data preprocessing techniques which help us to prepare the data for ML model building
- Analyze the dataset and extract some key information from the dataset (such as highest and lowest manufacturing plant location, country wise accident level count, sector wise accident level count, gender ratio, season wise accident level count etc.)
- Build the machine learning classifier and gained highest accuracy of 94% in SVM but there is an overfitting in training as we have received 100% accuracy on training. We will try to resolve the issue in the upcoming DL models.

OVERVIEW OF MILESTONE-2



The brief approach for the solution is given as follows for milestone-2:

- Design, train and test Neural networks classifiers and print the accuracy score of both train and test.
- Design, train and test RNN or LSTM classifiers and print the accuracy score of both train and test.
- Try bidirectional RNN and LSTM to see any changes in accuracy.
- Try different activation function and also increase epoch number.
- Choose the best model and pickle the model for future use.
- Create web service GUI using FLASK Library.

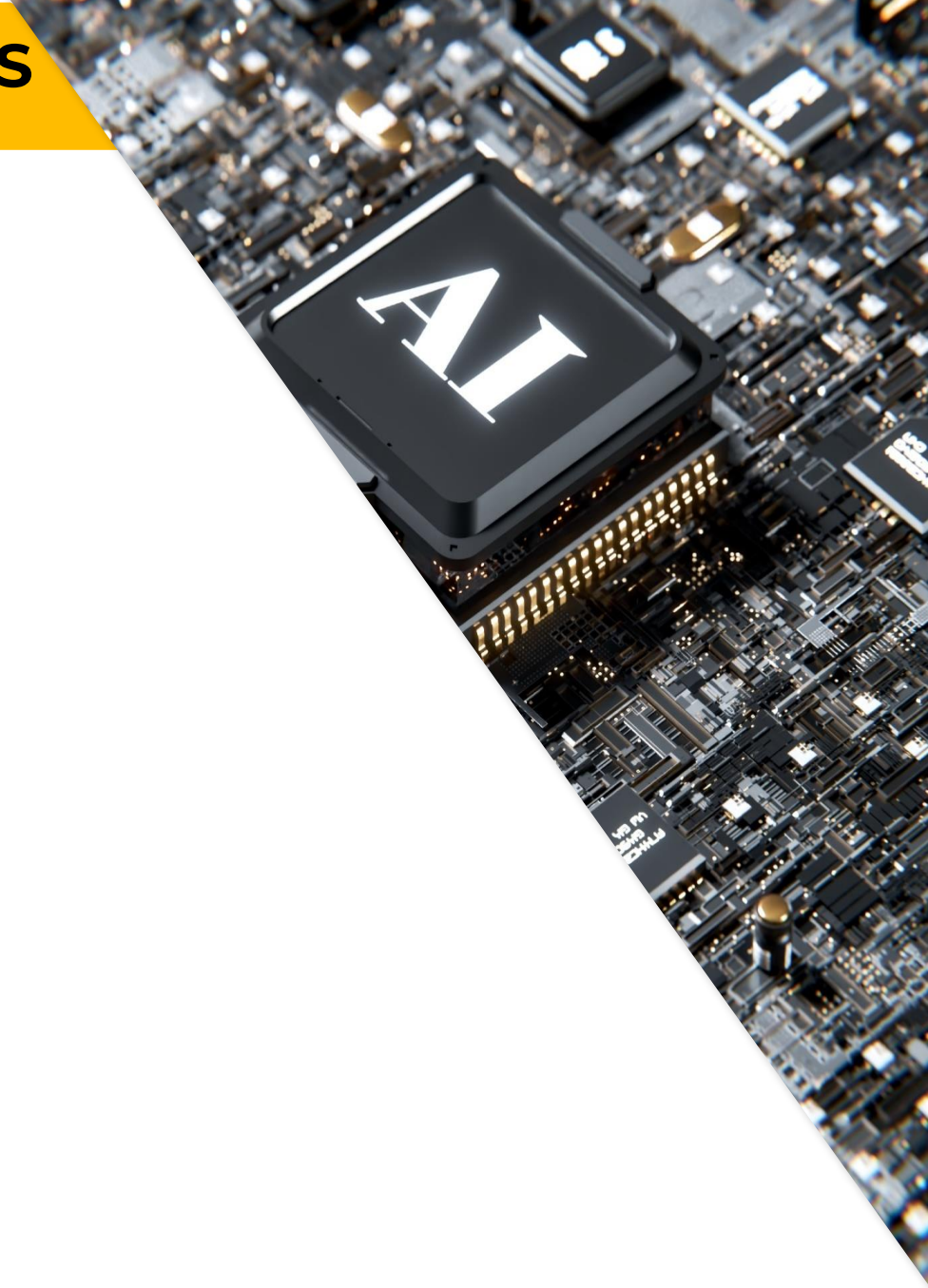
INSTALL SOME LIBRARIES BEFORE DL MODELS

We need to install or import some libraries as tensorflow, unidecode, autocorrect , google_trans_new, ann_visualizer and googletrans before running the deep learning models

```
import tensorflow as tf
tf.__version__
```

```
'2.15.0'
```

```
!pip install unidecode
!pip install autocorrect
!pip install google_trans_new
!pip install ann_visualizer
!pip install googletrans
```



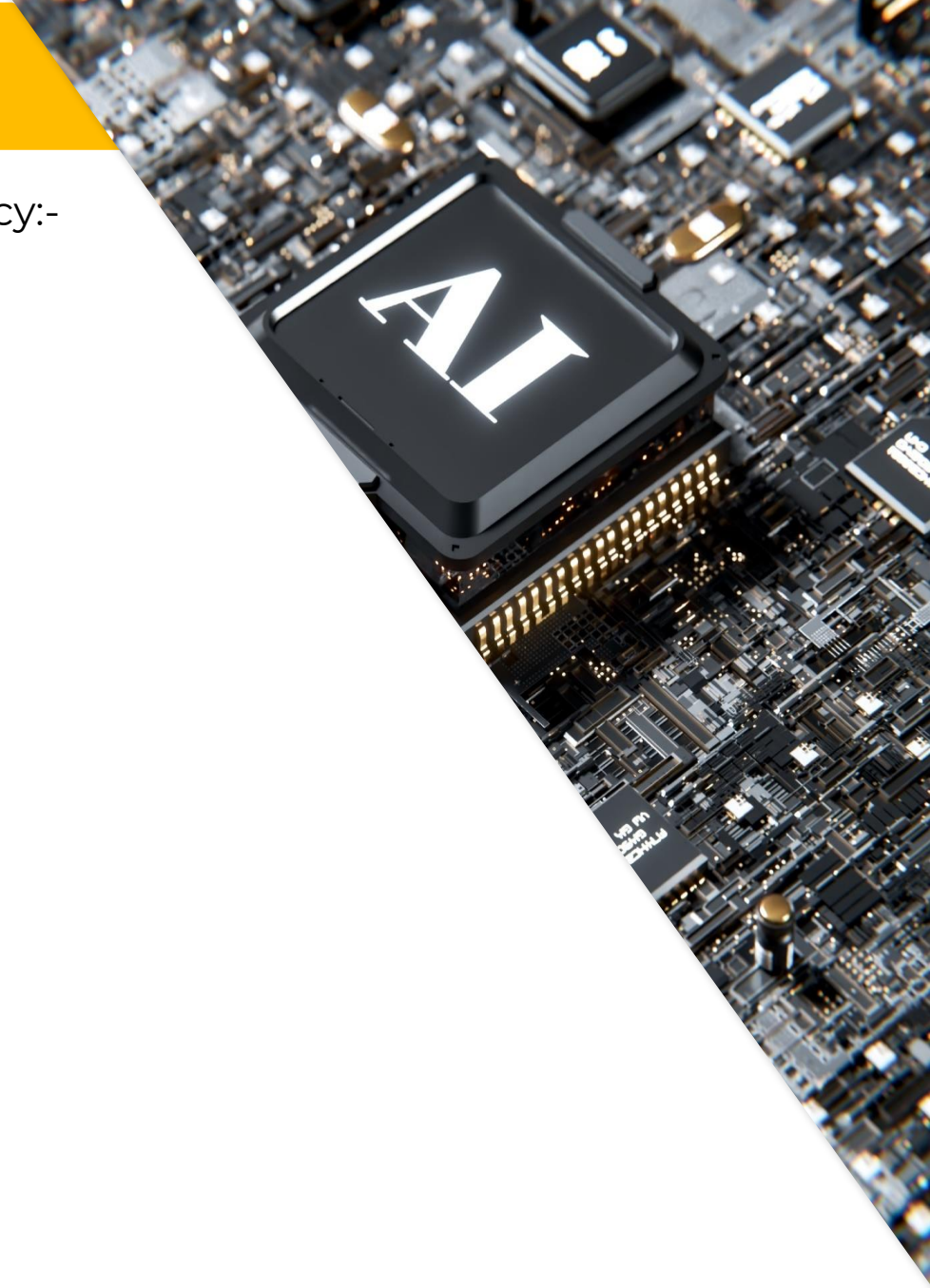
EXECUTE ANN CLASSIFIER

At first, let's execute the ANN classifier on the dataset and see the accuracy:-

ANN

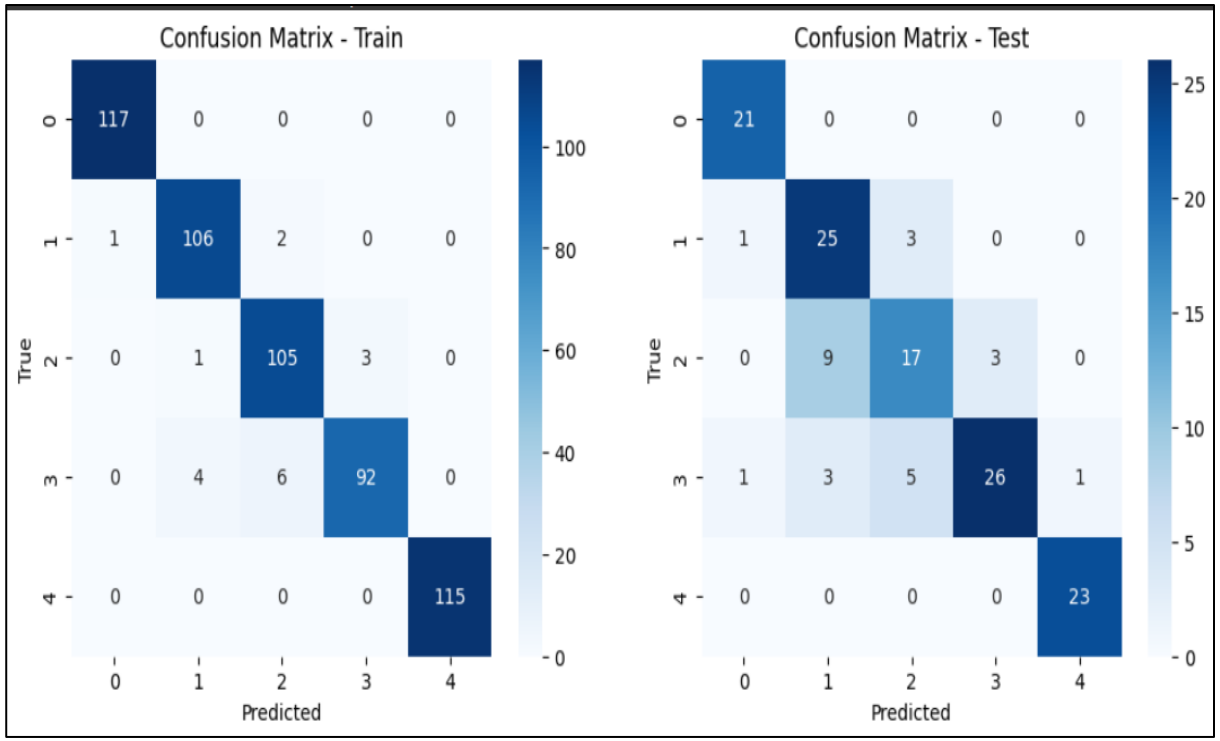
```
# @title ANN
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import LabelEncoder # Import LabelEncoder
from tensorflow.keras.models import Sequential
from keras.layers import Input
# Encode labels
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Define ANN model
model_ANN = Sequential()
model_ANN.add(Input(shape=(X_train_tf.shape[1],)))
model_ANN.add(tf.keras.layers.Dense(128, activation='relu'))
model_ANN.add(tf.keras.layers.Dense(64, activation='relu'))
model_ANN.add(tf.keras.layers.Dense(len(label_encoder.classes_), activation='softmax'))  # Output layer with number of classes

# Compile the model
model_ANN.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
model_ANN.fit(X_train_tf.toarray(), y_train_encoded, epochs=30, batch_size=32, validation_split=0.3)

# Predictions
y_train_pred_prob = model_ANN.predict(X_train_tf.toarray())
y_train_pred = np.argmax(y_train_pred_prob, axis=1)
y_test_pred_prob = model_ANN.predict(X_test_tf.toarray())
y_test_pred = np.argmax(y_test_pred_prob, axis=1)

# Confusion matrices
cm_train = confusion_matrix(y_train_encoded, y_train_pred)
cm_test = confusion_matrix(y_test_encoded, y_test_pred)

# Display confusion matrices
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
sns.heatmap(cm_train, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix - Train')
plt.xlabel('Predicted')
plt.ylabel('True')

plt.subplot(1, 2, 2)
sns.heatmap(cm_test, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix - Test')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
# Calculate metrics
train_accuracy_ann = accuracy_score(y_train_encoded, y_train_pred)
test_accuracy_ann = accuracy_score(y_test_encoded, y_test_pred)
f1_ann = f1_score(y_test_encoded, y_test_pred, average='weighted')
precision_ann = precision_score(y_test_encoded, y_test_pred, average='weighted')
recall_ann = recall_score(y_test_encoded, y_test_pred, average='weighted')

# Display metrics
metrics_df = pd.DataFrame({
    'Metric': ['Train Accuracy_ANN', 'Test Accuracy_ANN', 'F1 Score_ANN', 'Precision_ANN', 'Recall_ANN'],
    'Value': [train_accuracy_ann, test_accuracy_ann, f1_ann, precision_ann, recall_ann]
})
print(metrics_df)
```



EXECUTE ANN CLASSIFIER

Executed ANN with 2 Relu hidden layers and Softmax as outer layer. Ran 30 epochs with 32 batch size.



	Metric	Value
0	Train Accuracy_ANN	0.969203
1	Test Accuracy_ANN	0.811594
2	F1 Score_ANN	0.808587
3	Precision_ANN	0.817435
4	Recall_ANN	0.811594

After running the ANN model, we have received 96% accuracy on training dataset and on test dataset, the accuracy is 81% with F1 score as 80%, precision and recall as 81%.

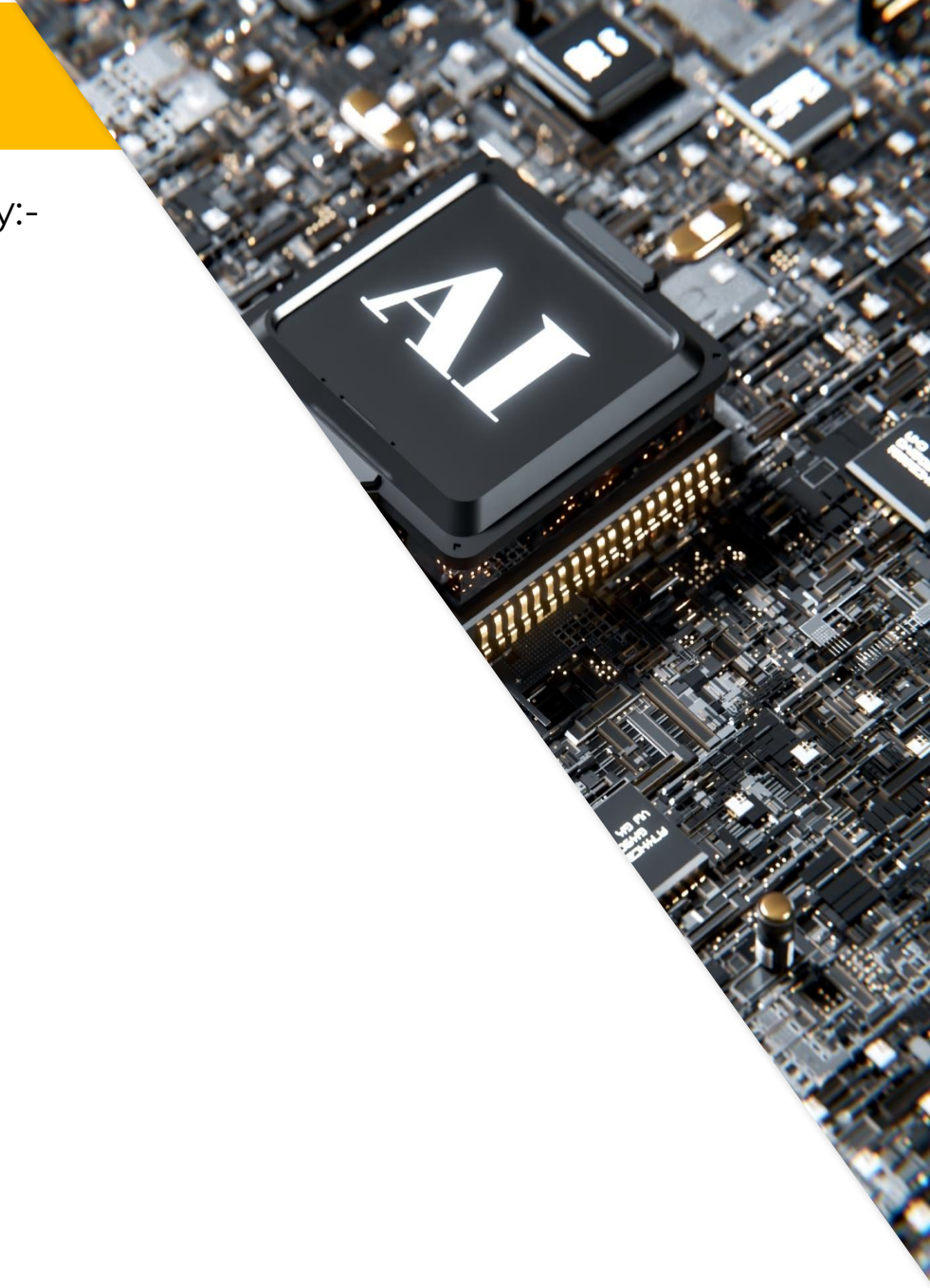
EXECUTE BASIC RNN CLASSIFIER

Let’s execute the basic RNN classifier on the dataset and see the accuracy:-

Basic RNN

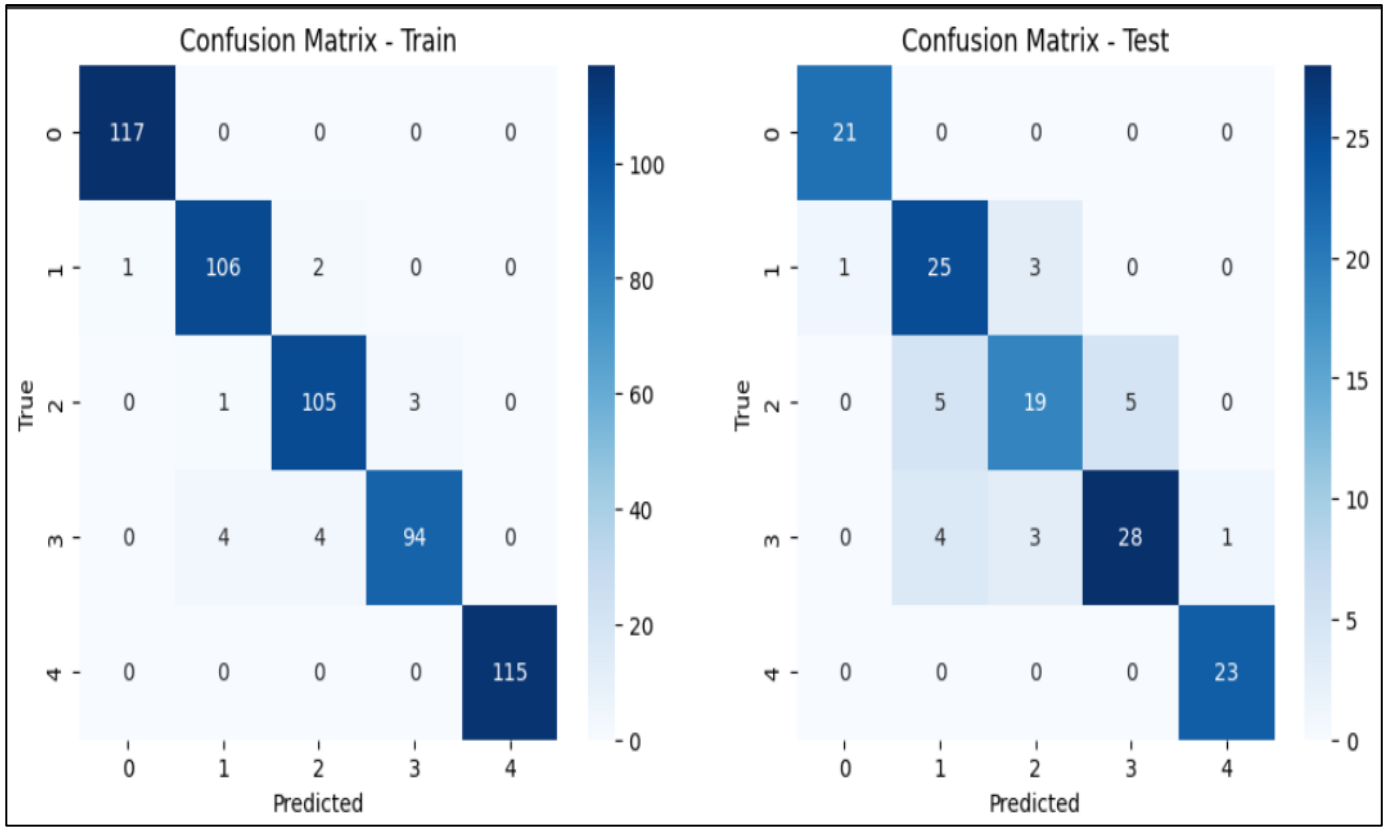
```
# @title Basic RNN
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import ConfusionMatrixDisplay

X_train_tf_reshaped = np.reshape(X_train_tf.toarray(), (X_train_tf.shape[0], 1, X_train_tf.shape[1]))
X_test_tf_reshaped = np.reshape(X_test_tf.toarray(), (X_test_tf.shape[0], 1, X_test_tf.shape[1]))

# Define RNN model
model_RNN = Sequential()
model_RNN.add(Input(shape=(X_train_tf_reshaped.shape[1], X_train_tf_reshaped.shape[2])))  # Update input shape
model_RNN.add(tf.keras.layers.SimpleRNN(512, activation='relu', return_sequences=True)) #Force the first RNN layer to return the full sequence
model_RNN.add(tf.keras.layers.SimpleRNN(128, activation='relu'))
model_RNN.add(tf.keras.layers.Dense(64, activation='relu'))
model_RNN.add(tf.keras.layers.Dense(len(label_encoder.classes_), activation='softmax'))

# Compile the model
model_RNN.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
model_RNN.fit(X_train_tf_reshaped, y_train_encoded, epochs=10, batch_size=32, validation_split=0.2)

# Predictions
y_train_pred_prob = model_RNN.predict(X_train_tf_reshaped)
y_train_pred = np.argmax(y_train_pred_prob, axis=1)
y_test_pred_prob = model_RNN.predict(X_test_tf_reshaped)
y_test_pred = np.argmax(y_test_pred_prob, axis=1)

# Confusion matrices
cm_train = confusion_matrix(y_train_encoded, y_train_pred)
cm_test = confusion_matrix(y_test_encoded, y_test_pred)

# Display confusion matrices
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
sns.heatmap(cm_train, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix - Train')
plt.xlabel('Predicted')
plt.ylabel('True')

plt.subplot(1, 2, 2)
sns.heatmap(cm_test, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix - Test')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
# Calculate metrics
train_accuracy_RNN = accuracy_score(y_train_encoded, y_train_pred)
test_accuracy_RNN = accuracy_score(y_test_encoded, y_test_pred)
f1_RNN = f1_score(y_test_encoded, y_test_pred, average='weighted')
precision_RNN = precision_score(y_test_encoded, y_test_pred, average='weighted')
recall_RNN = recall_score(y_test_encoded, y_test_pred, average='weighted')

# Display metrics
metrics_df = pd.DataFrame({
    'Metric': ['Train Accuracy_RNN', 'Test Accuracy_RNN', 'F1 Score_RNN', 'Precision_RNN', 'Recall_RNN'],
    'Value': [train_accuracy_RNN, test_accuracy_RNN, f1_RNN, precision_RNN, recall_RNN]
})
print(metrics_df)
```



EXECUTE BASIC RNN CLASSIFIER

Executed basic RNN with 3 Relu hidden layers and Softmax as outer layer.
Ran 10 epochs with 32 batch size.



	Metric	Value
0	Train Accuracy_RNN	0.972826
1	Test Accuracy_RNN	0.840580
2	F1 Score_RNN	0.838137
3	Precision_RNN	0.840551
4	Recall_RNN	0.840580

After running the Basic RNN model, we have received 97% accuracy on training dataset and on test dataset, the accuracy is 84% with F1 score as 84 %, precision and recall as 84%.

EXECUTE BI-DIRECTIONAL RNN CLASSIFIER

Let’s execute the Bi-directional RNN classifier on the dataset and see the accuracy:-

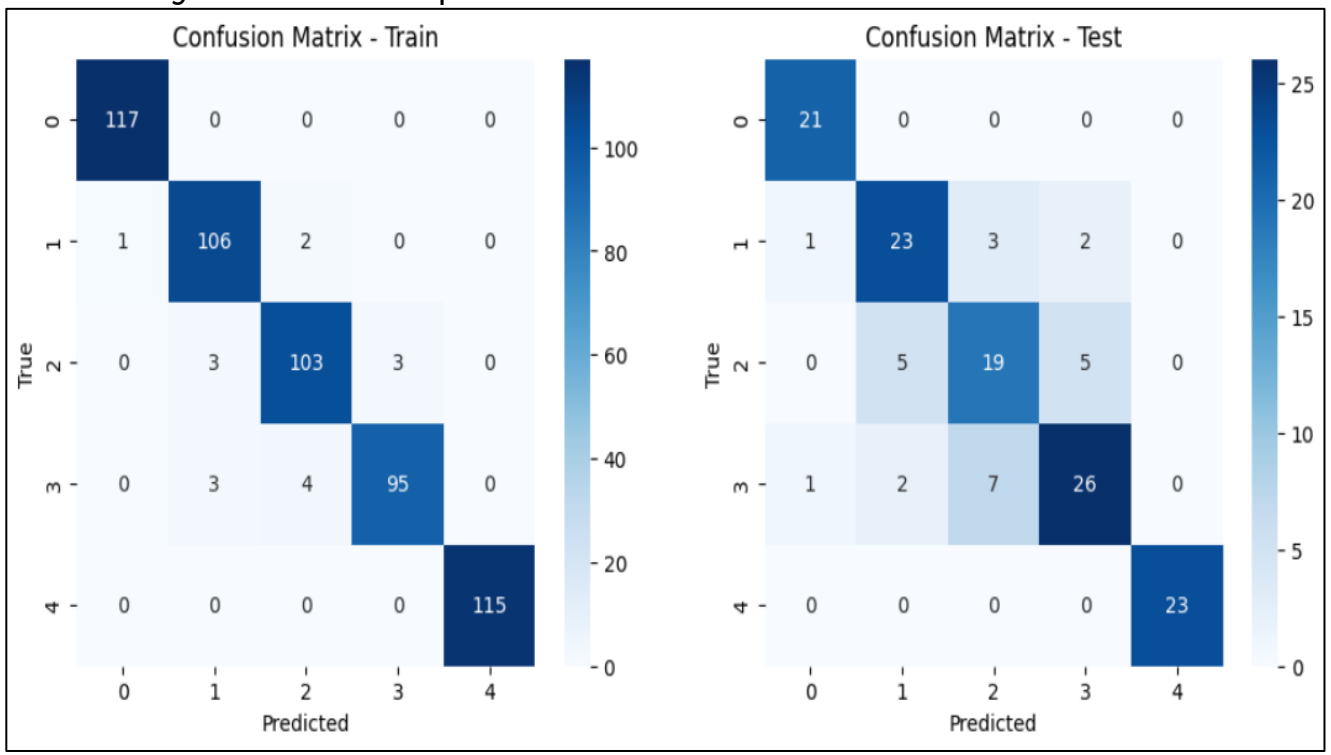
Bi-directional RNN

```
# @title Bi-directional RNN
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
# Reshape data for Bidirectional RNN
X_train_tf_reshaped = np.reshape(X_train_tf.toarray(), (X_train_tf.shape[0], 1, X_train_tf.shape[1]))
X_test_tf_reshaped = np.reshape(X_test_tf.toarray(), (X_test_tf.shape[0], 1, X_test_tf.shape[1]))

# Define Bidirectional RNN model
model_BiRNN = Sequential()
model_BiRNN.add(Input(shape=(X_train_tf_reshaped.shape[1], X_train_tf_reshaped.shape[2])))
model_BiRNN.add(tf.keras.layers.Bidirectional(tf.keras.layers.SimpleRNN(512, activation='relu', return_sequences=True))) # Add return_sequences=True
model_BiRNN.add(tf.keras.layers.Bidirectional(tf.keras.layers.SimpleRNN(128, activation='relu')))
model_BiRNN.add(tf.keras.layers.Dense(64, activation='relu'))
model_BiRNN.add(tf.keras.layers.Dense(len(label_encoder.classes_), activation='softmax'))

# Compile the model
model_BiRNN.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
model_BiRNN.fit(X_train_tf_reshaped, y_train_encoded, epochs=10, batch_size=32, validation_split=0.2)

# Predictions
y_train_pred_prob = model_BiRNN.predict(X_train_tf_reshaped)
y_train_pred = np.argmax(y_train_pred_prob, axis=1)
y_test_pred_prob = model_BiRNN.predict(X_test_tf_reshaped)
y_test_pred = np.argmax(y_test_pred_prob, axis=1)

# Confusion matrices
cm_train = confusion_matrix(y_train_encoded, y_train_pred)
cm_test = confusion_matrix(y_test_encoded, y_test_pred)

# Display confusion matrices
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
sns.heatmap(cm_train, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix - Train')
plt.xlabel('Predicted')
plt.ylabel('True')

plt.subplot(1, 2, 2)
sns.heatmap(cm_test, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix - Test')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
# Calculate metrics
train_accuracy_BiRNN = accuracy_score(y_train_encoded, y_train_pred)
test_accuracy_BiRNN = accuracy_score(y_test_encoded, y_test_pred)
f1_BiRNN = f1_score(y_test_encoded, y_test_pred, average='weighted')
precision_BiRNN = precision_score(y_test_encoded, y_test_pred, average='weighted')
recall_BiRNN = recall_score(y_test_encoded, y_test_pred, average='weighted')

# Display metrics
metrics_df = pd.DataFrame({
    'Metric': ['Train Accuracy_BiRNN', 'Test Accuracy_BiRNN', 'F1 Score_BiRNN', 'Precision_BiRNN', 'Recall_BiRNN'],
    'Value': [train_accuracy_BiRNN, test_accuracy_BiRNN, f1_BiRNN, precision_BiRNN, recall_BiRNN]
})
print(metrics_df)
```

EXECUTE BI-DIRECTIONAL RNN CLASSIFIER

Executed Bi-directional RNN with 3 Relu hidden layers and Softmax as outer layer. Ran 10 epochs with 32 batch size.



	Metric	Value
0	Train Accuracy_BiRNN	0.971014
1	Test Accuracy_BiRNN	0.811594
2	F1 Score_BiRNN	0.810044
3	Precision_BiRNN	0.809934
4	Recall_BiRNN	0.811594

After running the bi-directional RNN model, we have received 97% accuracy on training dataset and on test dataset, the accuracy is 81% with F1 score as 81 %, precision and recall as 81 %.

EXECUTE BASIC LSTM CLASSIFIER

Let’s execute the Basic LSTM classifier on the dataset and see the accuracy:-

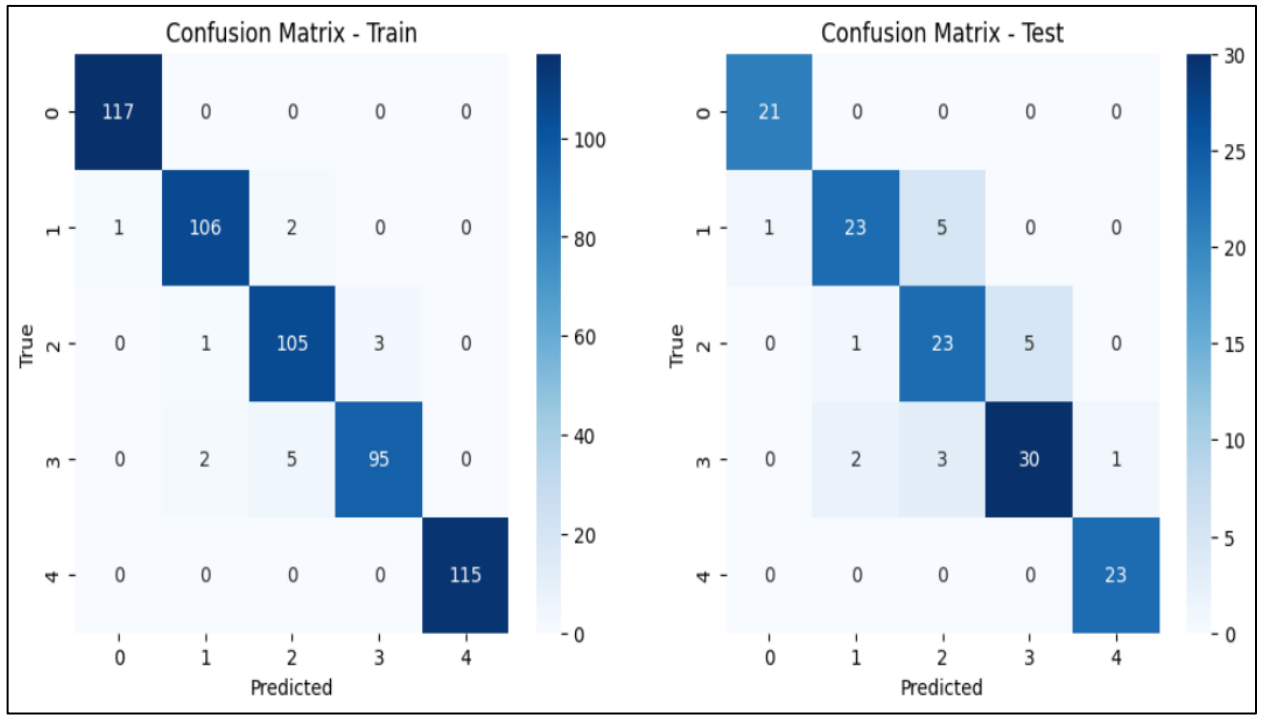
Normal LSTM

```
[ ] # @title Normal LSTM
    import pandas as pd
    import matplotlib.pyplot as plt
    import numpy as np
    # Reshape data for LSTM
    X_train_tf_reshaped = np.reshape(X_train_tf.toarray(), (X_train_tf.shape[0], 1, X_train_tf.shape[1]))
    X_test_tf_reshaped = np.reshape(X_test_tf.toarray(), (X_test_tf.shape[0], 1, X_test_tf.shape[1]))

    # Define LSTM model
    model_LSTM = Sequential()
    model_LSTM.add(Input(shape=(X_train_tf_reshaped.shape[1], X_train_tf_reshaped.shape[2])))
    model_LSTM.add(tf.keras.layers.LSTM(512, activation='relu', return_sequences=True))
    model_LSTM.add(tf.keras.layers.LSTM(128, activation='relu'))
    model_LSTM.add(tf.keras.layers.Dense(64, activation='relu'))
    model_LSTM.add(tf.keras.layers.Dense(len(label_encoder.classes_), activation='softmax'))

    # Compile the model
    model_LSTM.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

    # Train the model
    model_LSTM.fit(X_train_tf_reshaped, y_train_encoded, epochs=10, batch_size=32, validation_split=0.2)

    # Predictions
    y_train_pred_prob = model_LSTM.predict(X_train_tf_reshaped)
    y_train_pred = np.argmax(y_train_pred_prob, axis=1)
    y_test_pred_prob = model_LSTM.predict(X_test_tf_reshaped)
    y_test_pred = np.argmax(y_test_pred_prob, axis=1)

    # Confusion matrices
    cm_train = confusion_matrix(y_train_encoded, y_train_pred)
    cm_test = confusion_matrix(y_test_encoded, y_test_pred)

    # Display confusion matrices
    plt.figure(figsize=(12, 5))
    plt.subplot(1, 2, 1)
    sns.heatmap(cm_train, annot=True, fmt='d', cmap='Blues')
    plt.title('Confusion Matrix - Train')
    plt.xlabel('Predicted')
    plt.ylabel('True')

    plt.subplot(1, 2, 2)
    sns.heatmap(cm_test, annot=True, fmt='d', cmap='Blues')
    plt.title('Confusion Matrix - Test')
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.show()
```

```
# Calculate metrics
train_accuracy_LSTM = accuracy_score(y_train_encoded, y_train_pred)
test_accuracy_LSTM = accuracy_score(y_test_encoded, y_test_pred)
f1_LSTM = f1_score(y_test_encoded, y_test_pred, average='weighted')
precision_LSTM = precision_score(y_test_encoded, y_test_pred, average='weighted')
recall_LSTM = recall_score(y_test_encoded, y_test_pred, average='weighted')

# Display metrics
metrics_df = pd.DataFrame({
    'Metric': ['Train Accuracy_LSTM', 'Test Accuracy_LSTM', 'F1 Score_LSTM', 'Precision_LSTM', 'Recall_LSTM'],
    'Value': [train_accuracy_LSTM, test_accuracy_LSTM, f1_LSTM, precision_LSTM, recall_LSTM]
})
print(metrics_df)
```

EXECUTE BASIC LSTM CLASSIFIER

Executed Basic LSTM with 3 Relu hidden layers and Softmax as outer layer.
Ran 10 epochs with 32 batch size.



	Metric	Value
0	Train Accuracy_LSTM	0.974638
1	Test Accuracy_LSTM	0.869565
2	F1 Score_LSTM	0.869077
3	Precision_LSTM	0.870393
4	Recall_LSTM	0.869565

After running the Basic LSTM model, we have received 97% accuracy on training dataset and on test dataset, the accuracy is 86% with F1 score as 86 %, precision and recall as 86 %.

EXECUTE BI-DIRECTIONAL LSTM CLASSIFIER

Let’s execute the Bi-directional LSTM classifier on the dataset and see the accuracy:-

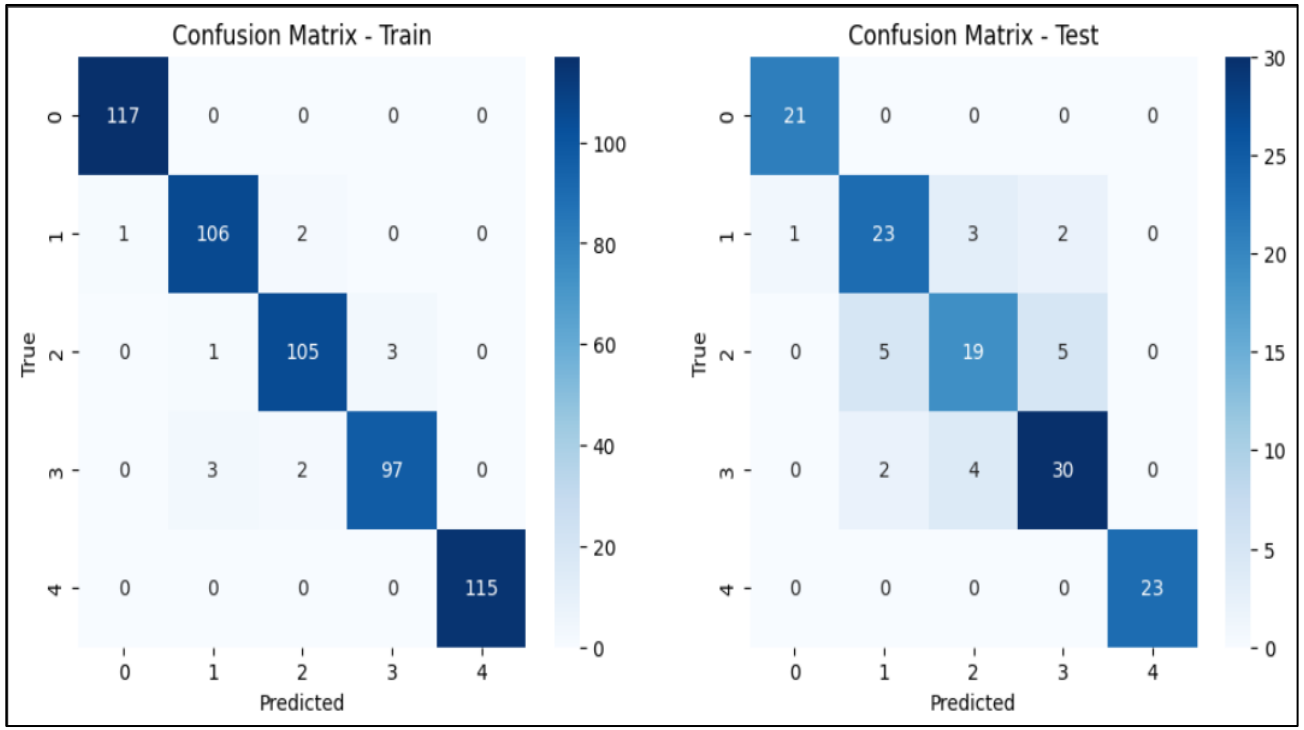
Bi-directional LSTM

```
# @title Bi-directional LSTM
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.optimizers import Adam
# Reshape data for Bidirectional LSTM
X_train_tf_reshaped = np.reshape(X_train_tf.toarray(), (X_train_tf.shape[0], 1, X_train_tf.shape[1]))
X_test_tf_reshaped = np.reshape(X_test_tf.toarray(), (X_test_tf.shape[0], 1, X_test_tf.shape[1]))

# Define Bidirectional LSTM model
model_BiLSTM = Sequential()
model_BiLSTM.add(Input(shape=(X_train_tf_reshaped.shape[1], X_train_tf_reshaped.shape[2])))
model_BiLSTM.add(Bidirectional(LSTM(1024, activation='relu', return_sequences=True)))
model_BiLSTM.add(Bidirectional(LSTM(512, activation='relu', return_sequences=True)))
model_BiLSTM.add(Bidirectional(LSTM(256, activation='relu', return_sequences=True)))
model_BiLSTM.add(Bidirectional(LSTM(128, activation='relu')))
model_BiLSTM.add(Dense(64, activation='relu'))
model_BiLSTM.add(Dropout(0.5))  # Add dropout layer with 20% dropout rate
model_BiLSTM.add(Dense(len(label_encoder.classes_), activation='softmax'))

# Compile the model
learning_rate = 0.001  # Set your desired learning rate
optimizer = Adam(learning_rate=learning_rate)
model_BiLSTM.compile(optimizer=optimizer, loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
model_BiLSTM.fit(X_train_tf_reshaped, y_train_encoded, epochs=100, batch_size=64, validation_split=0.2)

# Predictions
y_train_pred_prob = model_BiLSTM.predict(X_train_tf_reshaped)
y_train_pred = np.argmax(y_train_pred_prob, axis=1)
y_test_pred_prob = model_BiLSTM.predict(X_test_tf_reshaped)
y_test_pred = np.argmax(y_test_pred_prob, axis=1)

# Confusion matrices
cm_train = confusion_matrix(y_train_encoded, y_train_pred)
cm_test = confusion_matrix(y_test_encoded, y_test_pred)

# Display confusion matrices
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
sns.heatmap(cm_train, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix - Train')
plt.xlabel('Predicted')
plt.ylabel('True')

plt.subplot(1, 2, 2)
sns.heatmap(cm_test, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix - Test')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
# Calculate metrics
train_accuracy_BiLSTM = accuracy_score(y_train_encoded, y_train_pred)
test_accuracy_BiLSTM = accuracy_score(y_test_encoded, y_test_pred)
f1_BiLSTM = f1_score(y_test_encoded, y_test_pred, average='weighted')
precision_BiLSTM = precision_score(y_test_encoded, y_test_pred, average='weighted')
recall_BiLSTM = recall_score(y_test_encoded, y_test_pred, average='weighted')

# Display metrics
metrics_df = pd.DataFrame({
    'Metric': ['Train Accuracy_BiLSTM', 'Test Accuracy_BiLSTM', 'F1 Score_BiLSTM', 'Precision_BiLSTM', 'Recall_BiLSTM'],
    'Value': [train_accuracy_BiLSTM, test_accuracy_BiLSTM, f1_BiLSTM, precision_BiLSTM, recall_BiLSTM]
})
print(metrics_df)
```

EXECUTE BI-DIRECTIONAL LSTM CLASSIFIER

Executed Basic LSTM with 3 Relu hidden layers and Softmax as outer layer.
Ran 10 epochs with 32 batch size.

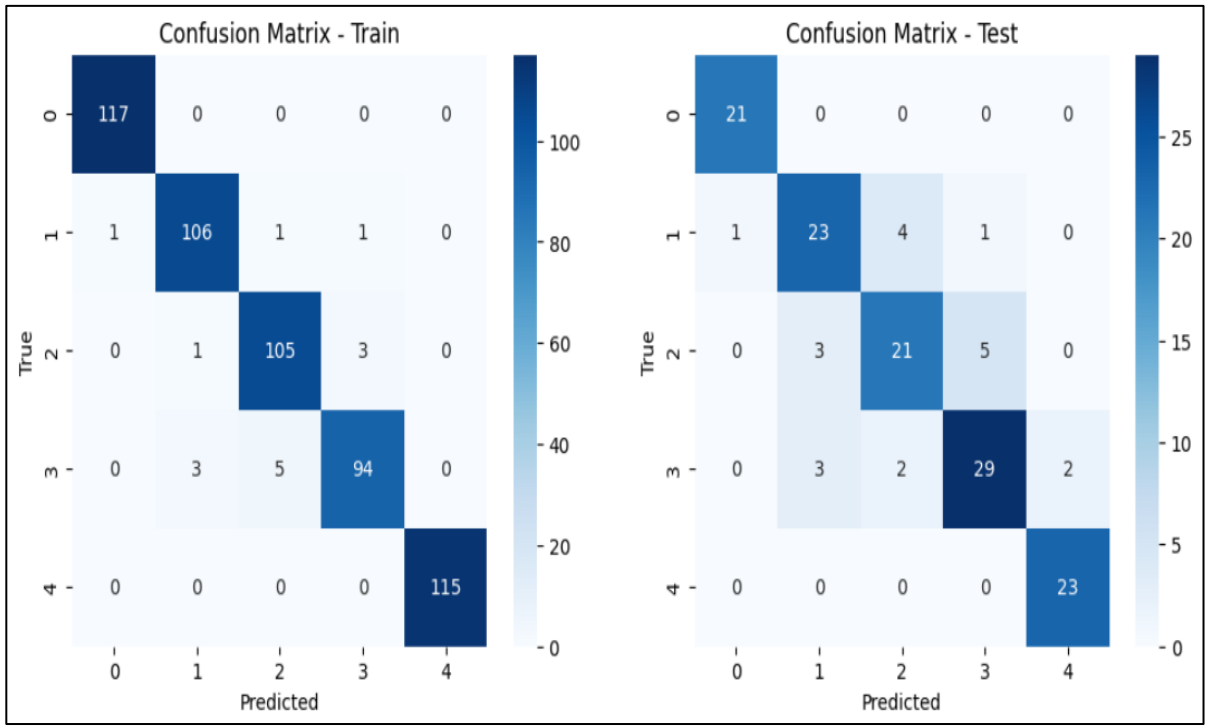


	Metric	Value
0	Train Accuracy_BiLSTM	0.978261
1	Test Accuracy_BiLSTM	0.840580
2	F1 Score_BiLSTM	0.838748
3	Precision_BiLSTM	0.838118
4	Recall_BiLSTM	0.840580

After running the bi-directional LSTM model, we have received 97% accuracy on training dataset and on test dataset, the accuracy is 84% with F1 score as 84 %, precision and recall as 84 %.

EXECUTE BI-DIRECTIONAL LSTM CLASSIFIER

Since basic and bi-directional LSTM is giving same result. Let's change the activation function to Tanh and see if we get any visible changes in the accuracy :-



	Metric	Value
0	Train Accuracy_BiLSTM	0.972826
1	Test Accuracy_BiLSTM	0.847826
2	F1 Score_BiLSTM	0.845737
3	Precision_BiLSTM	0.844852
4	Recall_BiLSTM	0.847826

After running the bi-directional LSTM with Tanh as activation function, we have received 97% accuracy on training dataset and on test dataset, the accuracy is 84% with F1 score as 84 %, precision and recall as 84 %.

EXECUTE BI-DIRECTIONAL GRU CLASSIFIER

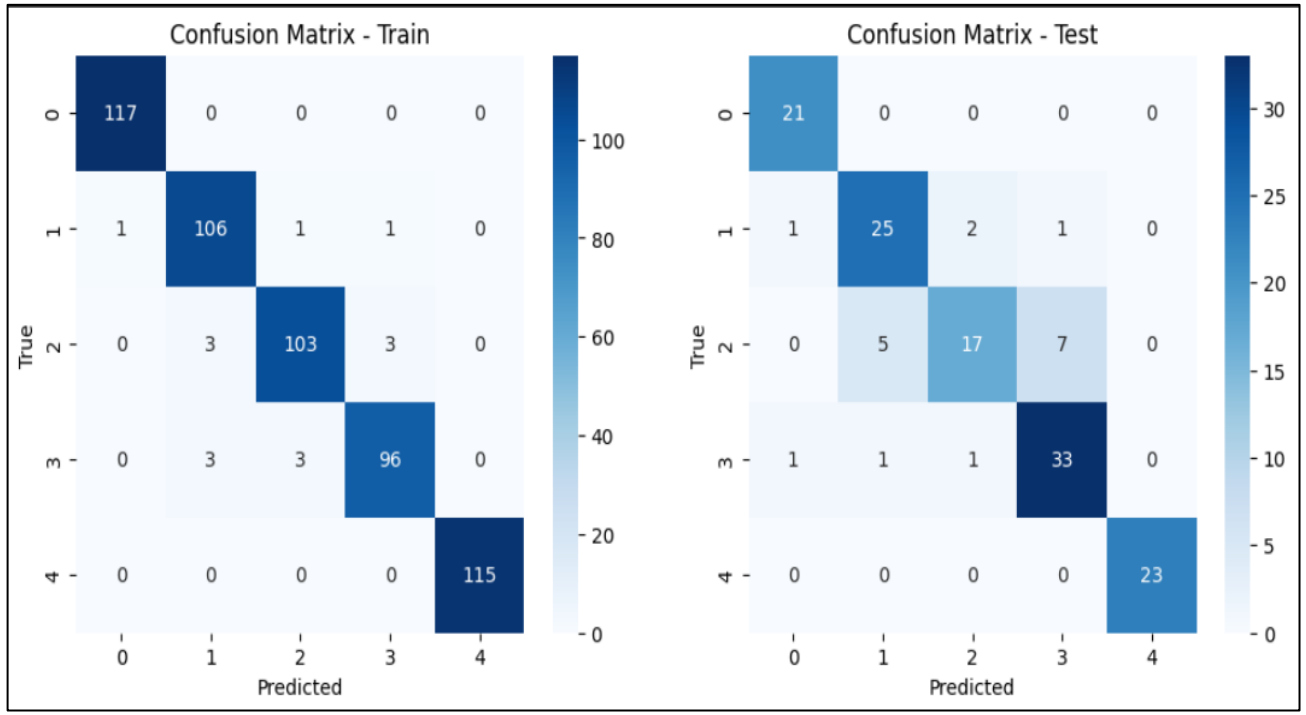
Let’s execute the Bi-directional GRU classifier on the dataset and see the accuracy:-

Bidirectional GRU

```
[ ] # @title Bidirectional GRU
    import pandas as pd
    import matplotlib.pyplot as plt
    import numpy as np
    # Reshape data for Bidirectional GRU
    X_train_tf_reshaped = np.reshape(X_train_tf.toarray(), (X_train_tf.shape[0], 1, X_train_tf.shape[1]))
    X_test_tf_reshaped = np.reshape(X_test_tf.toarray(), (X_test_tf.shape[0], 1, X_test_tf.shape[1]))

    # Define Bidirectional GRU model
    model_BiGRU = Sequential()
    model_BiGRU.add(Input(shape=(X_train_tf_reshaped.shape[1], X_train_tf_reshaped.shape[2])))
    model_BiGRU.add(tf.keras.layers.Bidirectional(tf.keras.layers.GRU(512, activation='relu', return_sequences=True)))
    model_BiGRU.add(tf.keras.layers.Bidirectional(tf.keras.layers.GRU(128, activation='relu')))
    model_BiGRU.add(tf.keras.layers.Dense(64, activation='relu'))
    model_BiGRU.add(tf.keras.layers.Dense(len(label_encoder.classes_), activation='softmax'))

    # Compile the model
    model_BiGRU.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

    # Train the model
    model_BiGRU.fit(X_train_tf_reshaped, y_train_encoded, epochs=10, batch_size=32, validation_split=0.2)

    # Predictions
    y_train_pred_prob = model_BiGRU.predict(X_train_tf_reshaped)
    y_train_pred = np.argmax(y_train_pred_prob, axis=1)
    y_test_pred_prob = model_BiGRU.predict(X_test_tf_reshaped)
    y_test_pred = np.argmax(y_test_pred_prob, axis=1)

    # Confusion matrices
    cm_train = confusion_matrix(y_train_encoded, y_train_pred)
    cm_test = confusion_matrix(y_test_encoded, y_test_pred)

    # Display confusion matrices
    plt.figure(figsize=(12, 5))
    plt.subplot(1, 2, 1)
    sns.heatmap(cm_train, annot=True, fmt='d', cmap='Blues')
    plt.title('Confusion Matrix - Train')
    plt.xlabel('Predicted')
    plt.ylabel('True')

    plt.subplot(1, 2, 2)
    sns.heatmap(cm_test, annot=True, fmt='d', cmap='Blues')
    plt.title('Confusion Matrix - Test')
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.show()
    # Calculate metrics
    train_accuracy_BiGRU = accuracy_score(y_train_encoded, y_train_pred)
    test_accuracy_BiGRU = accuracy_score(y_test_encoded, y_test_pred)
    f1_BiGRU = f1_score(y_test_encoded, y_test_pred, average='weighted')
    precision_BiGRU = precision_score(y_test_encoded, y_test_pred, average='weighted')
    recall_BiGRU = recall_score(y_test_encoded, y_test_pred, average='weighted')

    # Display metrics
    metrics_df = pd.DataFrame({
        'Metric': ['Train Accuracy_BiGRU', 'Test Accuracy_BiGRU', 'F1 Score_BiGRU', 'Precision_BiGRU', 'Recall_BiGRU'],
        'Value': [train_accuracy_BiGRU, test_accuracy_BiGRU, f1_BiGRU, precision_BiGRU, recall_BiGRU]
    })
    print(metrics_df)
```

EXECUTE BI-DIRECTIONAL GRU CLASSIFIER

Executed Bi-directional GRU with 3 Relu hidden layers and Softmax as outer layer.
Ran 10 epochs with 32 batch size.



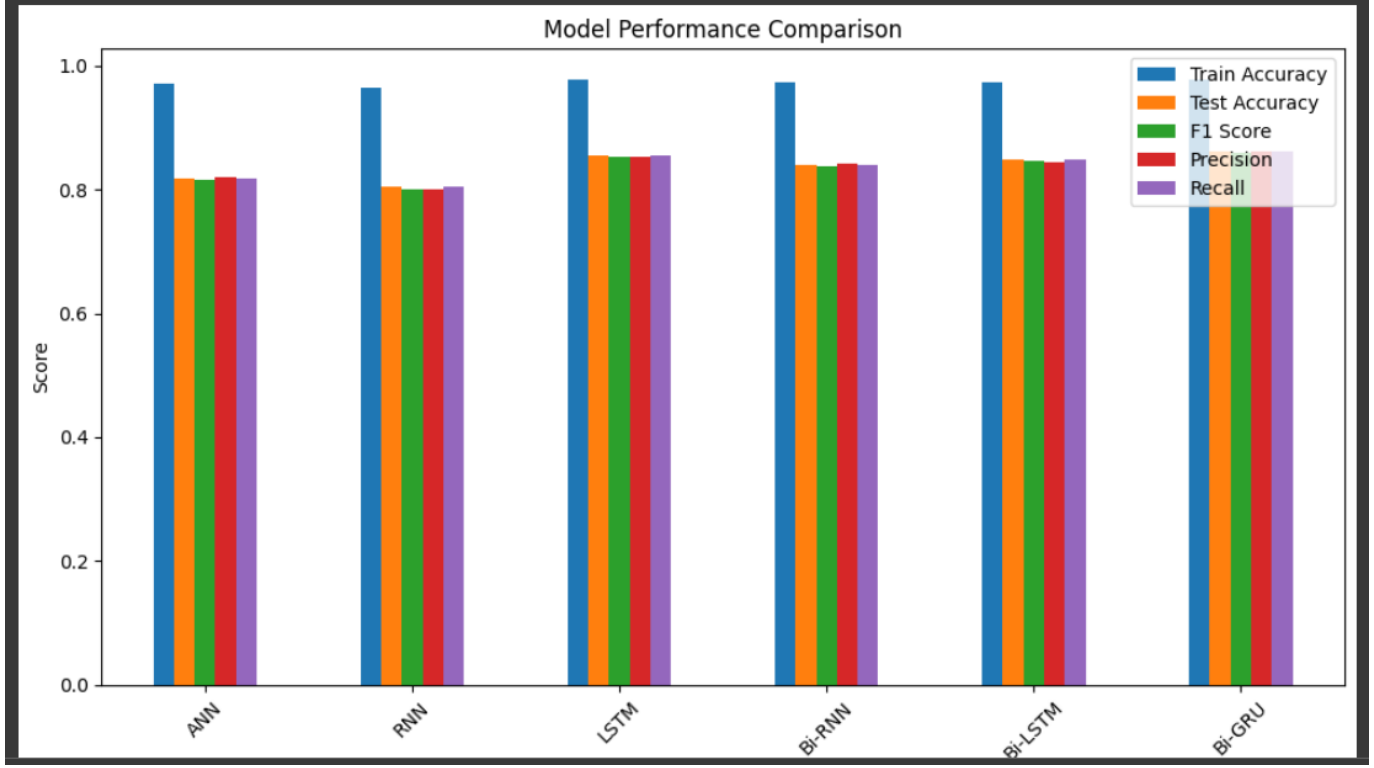
	Metric	Value
0	Train Accuracy_BiGRU	0.972826
1	Test Accuracy_BiGRU	0.862319
2	F1 Score_BiGRU	0.856462
3	Precision_BiGRU	0.863671
4	Recall_BiGRU	0.862319

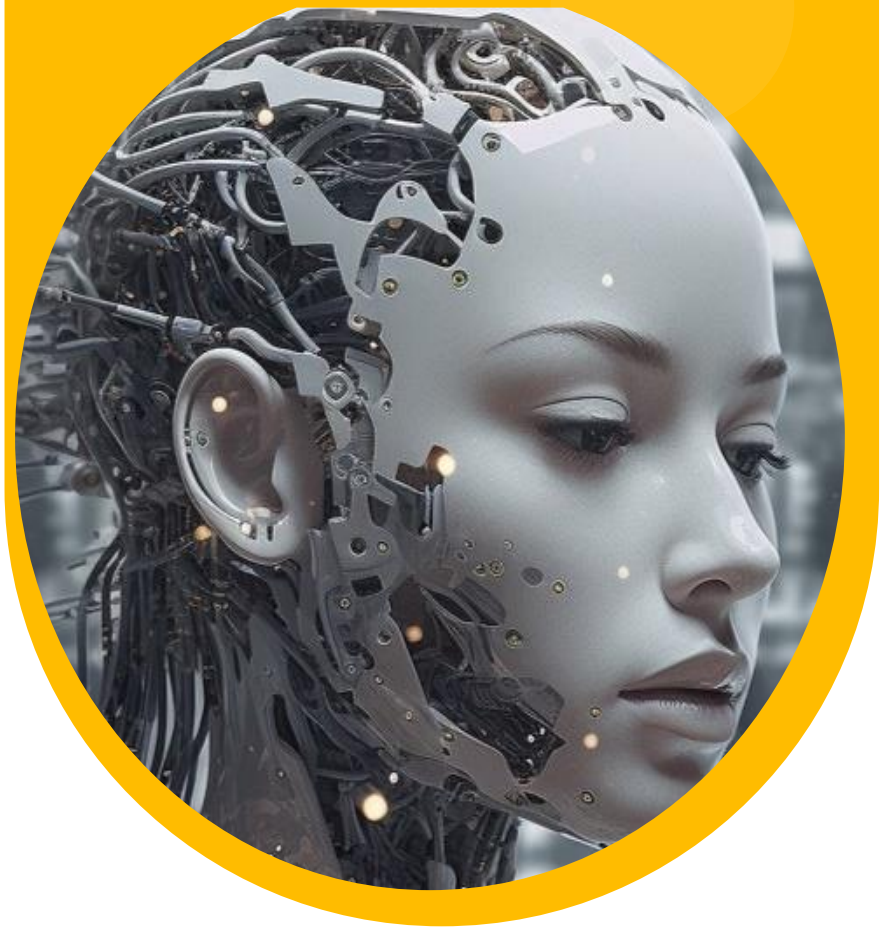
After running the bi-directional GRU model, we have received 97% accuracy on training dataset and on test dataset, the accuracy we got is 86% with F1 score as 85%, precision as 86% and recall as 86%.

MODEL BUILDING AND EVALUATION

Final result of all models :-

	Train Accuracy	Test Accuracy	F1 Score	Precision	Recall
ANN	0.971014	0.818841	0.815840	0.818961	0.818841
RNN	0.963768	0.804348	0.800344	0.801142	0.804348
LSTM	0.978261	0.855072	0.852422	0.853446	0.855072
Bi-RNN	0.972826	0.840580	0.836723	0.842552	0.840580
Bi-LSTM	0.972826	0.847826	0.845737	0.844852	0.847826
Bi-GRU	0.978261	0.862319	0.859165	0.860876	0.862319

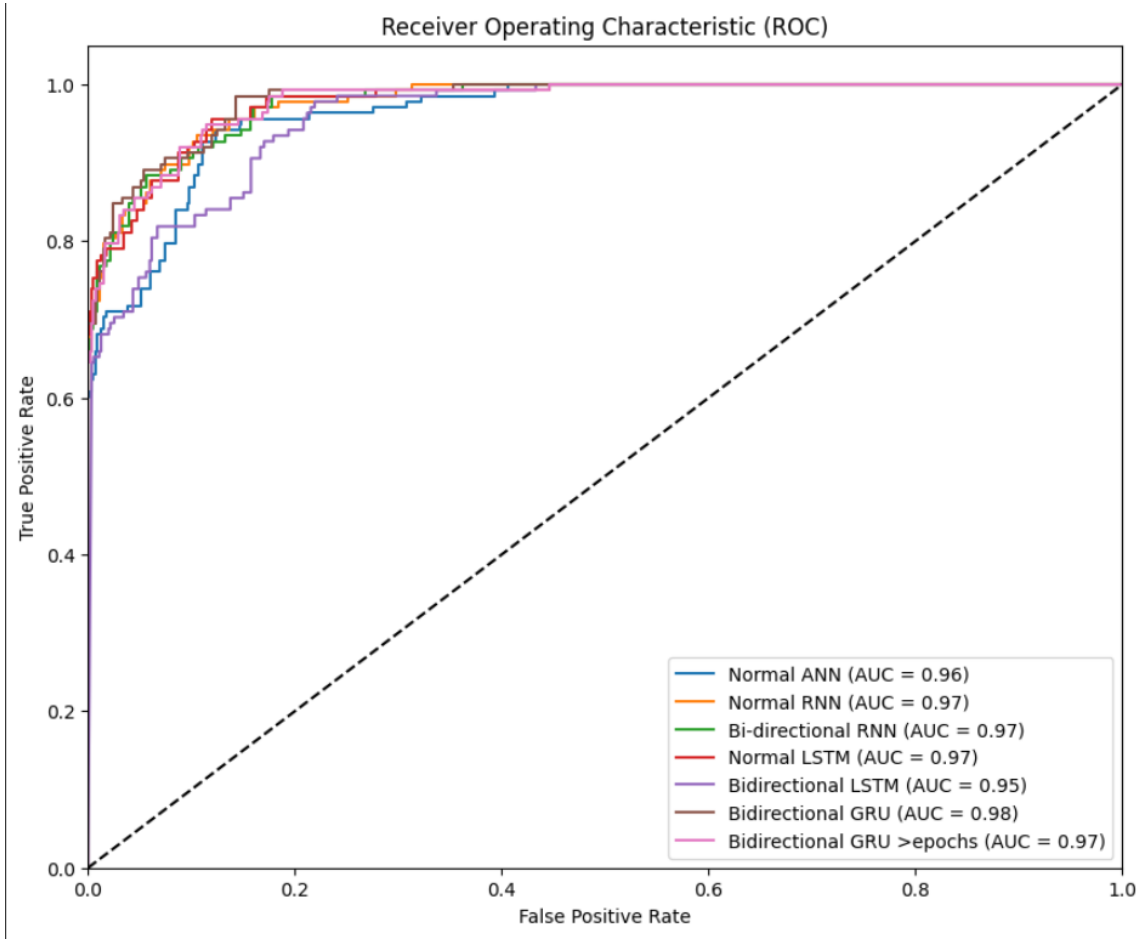


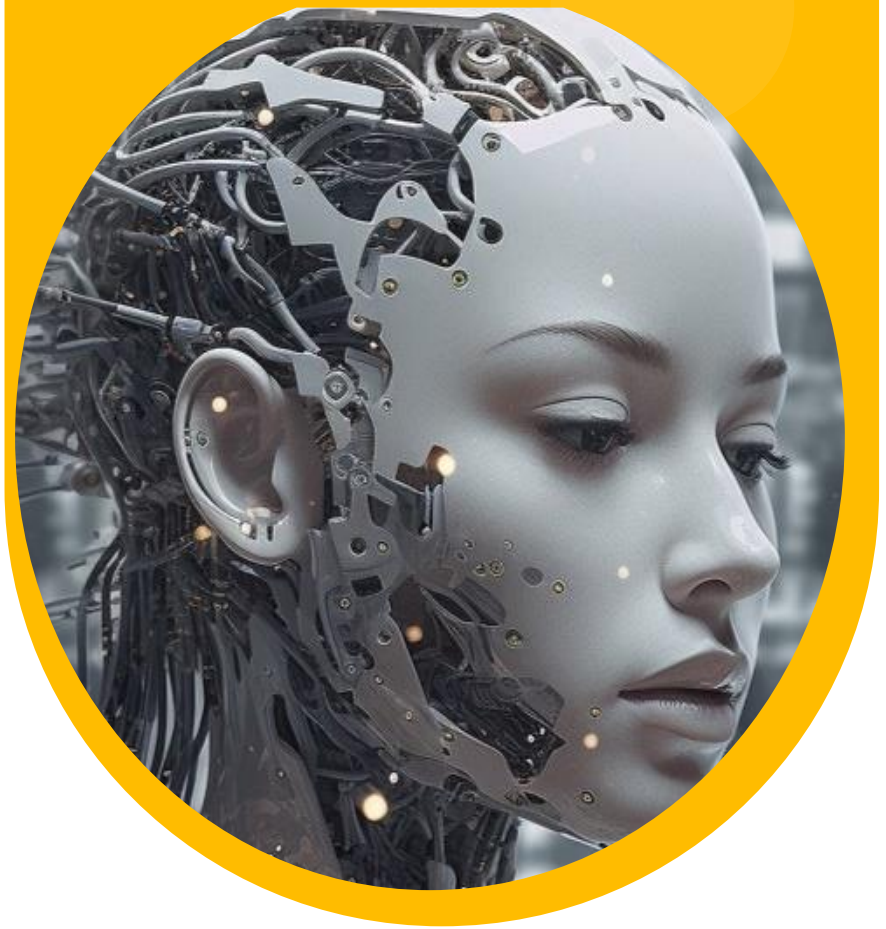


We can observe that Bi-directionalGRU provides best result in comparison to all the models. Hence, we can pickle Bi-directional RNN .

MODEL BUILDING AND EVALUATION

The ROC curve to analyze all model performance





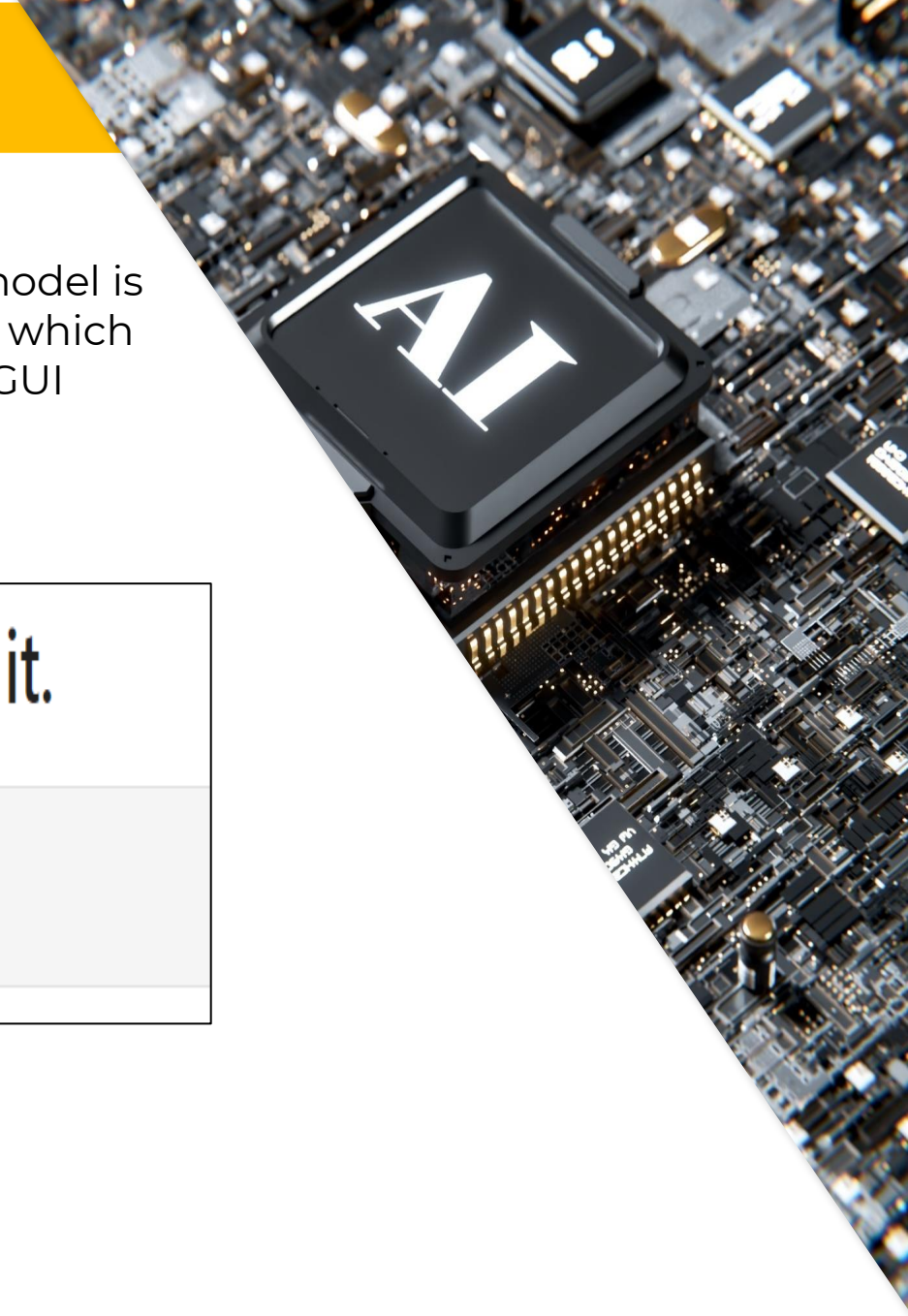
From the above ROC curve, we can see that Bi-directional GRU has best ROC curve in compared to other models

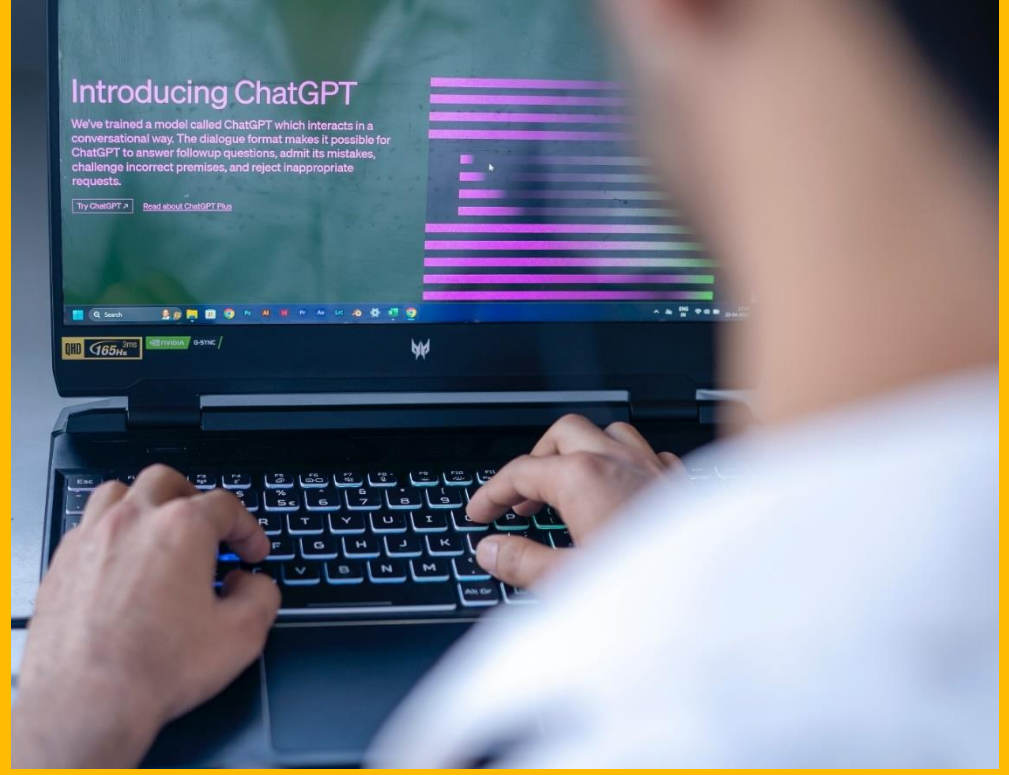
CHOOSE BEST MODEL

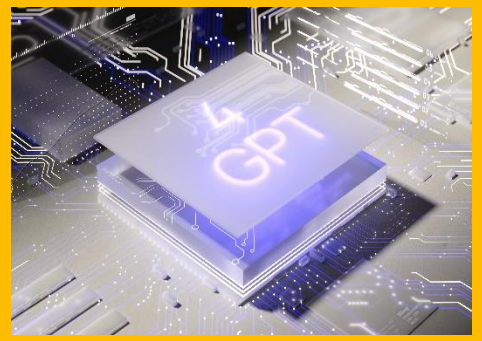
In all ML/DL models, SVM is getting best test accuracy, however, the SVM model is overfitted. Bi-directional GRU got balanced accuracy on both train and test which we can use in future. Hence, we have pickled Bi-directional GRU for future GUI building :-

Step 3: Choose the best performing classifier and pickle it.

In [ ]:
```
# Bi-directional has best accuracy. Hence, we have pickle the same for further use
filename = '/content/drive/MyDrive/Capstone Project/best_classifier.pkl'
pickle.dump(model_BiGRU, open(filename, 'wb'))
```







Milestone-2 Conclusion

Below are the steps we have taken in the milestone-2 to build a ML/DL based chatbot for the professionals to highlight the safety risk :-

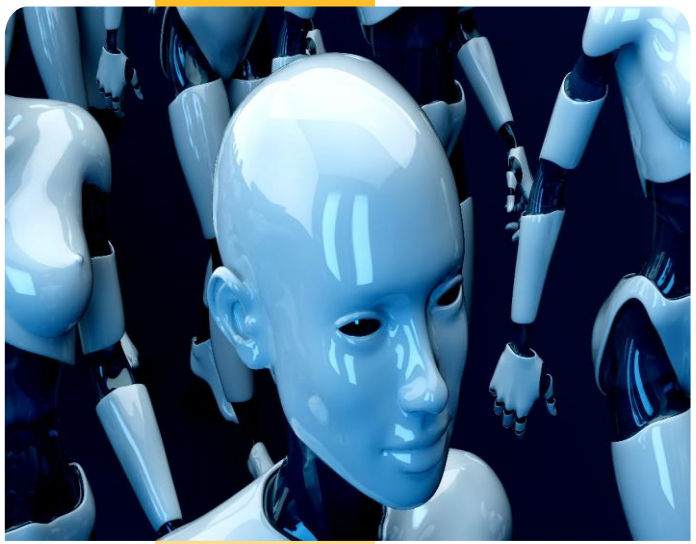
- We have executed the basic ANN, RNN and LSTM classifiers where we received the best train and test accuracy on LSTM model with train accuracy as 97% and test accuracy as 85%.

- We have tried Bi-directional RNN, LSTM and GRU where we received the best train and test accuracy on Bidirectional GRU model with train accuracy as 97% and test accuracy as 86%..

- We have tried to change the activation function from Relu to Tanh and also increased the number of epochs, but Bi-directional RNN still got best result.

- In all ML/DL models, SVM is getting best test accuracy, however, the SVM model is overfitted. Bi-directional GRU got balanced accuracy on both train and test which we can use in future. Hence, we have pickled Bi-directional GRU for future GUI building

OVERVIEW OF MILESTONE-3

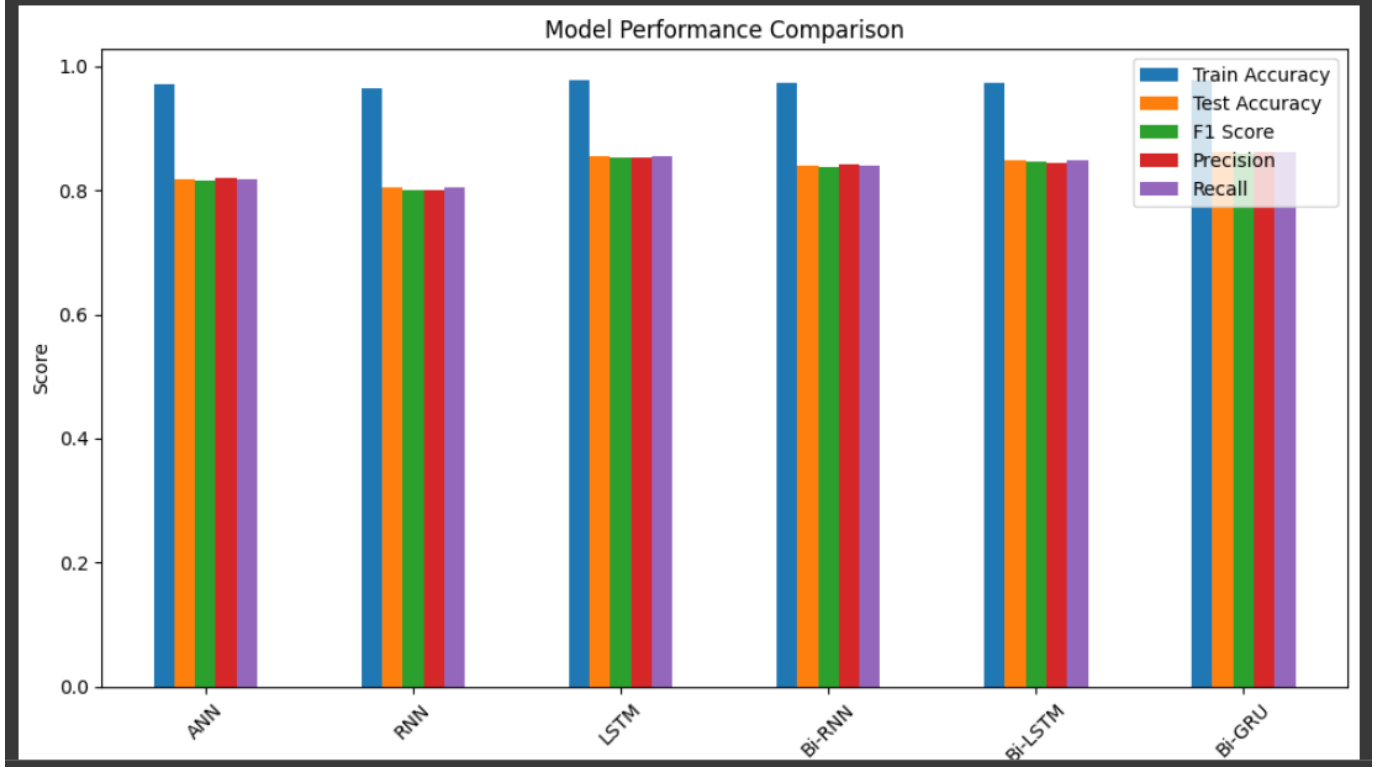


The brief approach for the solution is given as follows for milestone-3:

- Design a clickable UI based chatbot interface.

Final result of all models from milestone-2:-
It is evitable that we are using BiGRU as a model to further build our UI.

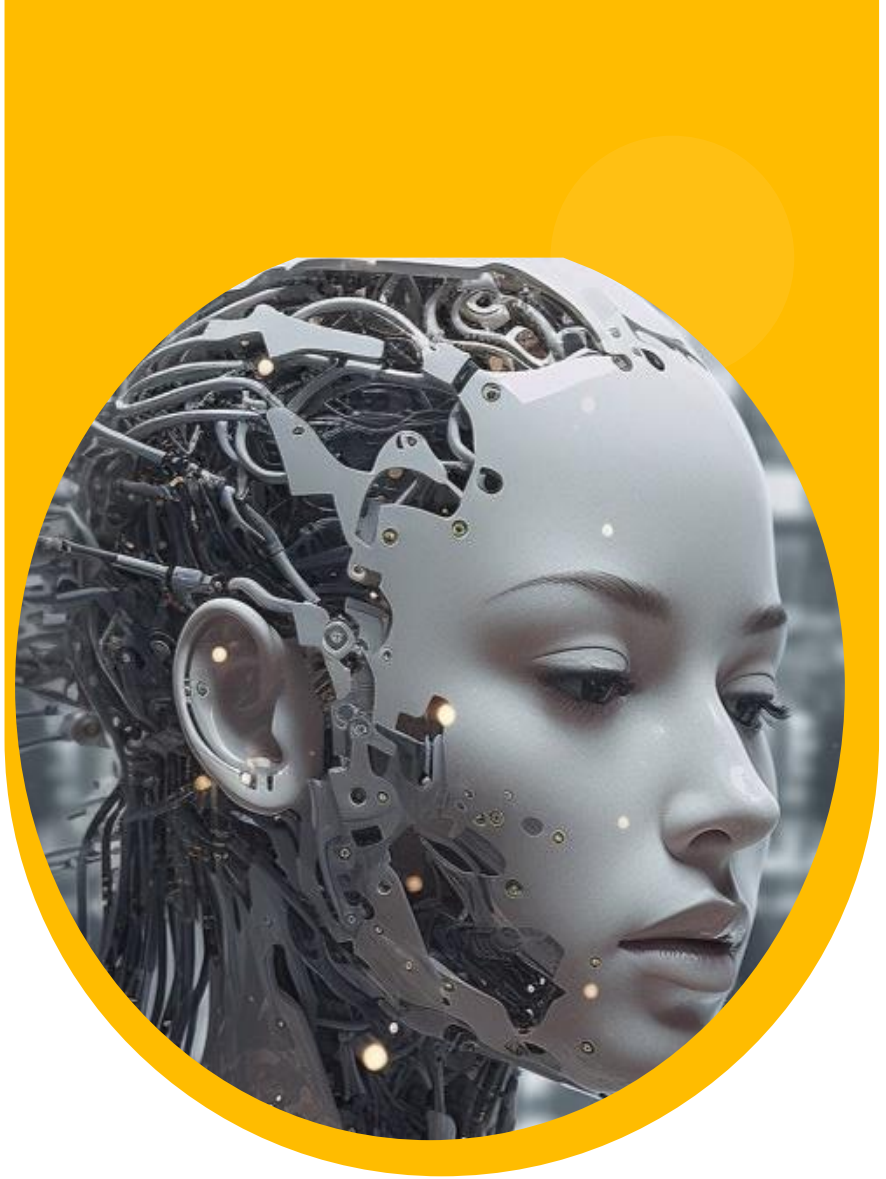
	Train Accuracy	Test Accuracy	F1 Score	Precision	Recall
ANN	0.971014	0.818841	0.815840	0.818961	0.818841
RNN	0.963768	0.804348	0.800344	0.801142	0.804348
LSTM	0.978261	0.855072	0.852422	0.853446	0.855072
Bi-RNN	0.972826	0.840580	0.836723	0.842552	0.840580
Bi-LSTM	0.972826	0.847826	0.845737	0.844852	0.847826
Bi-GRU	0.978261	0.862319	0.859165	0.860876	0.862319



CHOOSE BEST MODEL

We have observed from the ROC curve and summary of reports that best model in Bi-directional GRU. Hence, we have pickled the model for future use :-

```
# Assuming 'model_BiGRUepoch' is the best classifier based on your ROC AUC analysis
filename = '/content/drive/MyDrive/Capstone Project/best_classifier.pkl'
pickle.dump(model_BiGRU, open(filename, 'wb'))
```



Libraries imported for building the User Interface:

!pip install --ignore-installed blinker
!pip install --ignore-installed flask
import flask
!pip install pyngrok
!pip install pyngrok tensorflow Keras
import pickle
from flask import Flask, request, jsonify, render_template_string
import tensorflow as tf
import pickle
from keras.preprocessing.sequence import pad_sequences
from pyngrok import ngrok

Code for UI based on the Best Model

We have used Flask library and ngrok to create GUI for the chatbot

```
In [ ]: from flask import Flask, request, jsonify, render_template_string
        import tensorflow as tf
        import pickle
        from keras.preprocessing.sequence import pad_sequences
        from pyngrok import ngrok
        import nltk
        from nltk.tokenize import word_tokenize
        from nltk.corpus import stopwords
        from nltk.stem import WordNetLemmatizer
        import string

        nltk.download('punkt')
        nltk.download('stopwords')
        nltk.download('wordnet')

        # Initialize Flask app
        app = Flask(__name__)

        # Load model and tokenizer
        model = tf.keras.models.load_model('/content/drive/MyDrive/Capstone Project/model_BiGRU.h5')
        with open('/content/drive/MyDrive/Capstone Project/tokenizer.pkl', 'rb') as handle:
            tokenizer = pickle.load(handle)

        # Define preprocessing functions
        max_sequence_length = 1000

        def clean_text(text):
            tokens = word_tokenize(text.lower())
            table = str.maketrans('', '', string.punctuation)
            stripped_tokens = [w.translate(table) for w in tokens]
            words = [word for word in stripped_tokens if word.isalpha()]
            stop_words = set(stopwords.words('english'))
            words = [word for word in words if word not in stop_words]
            lemmatizer = WordNetLemmatizer()
            lemmatized = [lemmatizer.lemmatize(word) for word in words]
            return ' '.join(lemmatized)

        def preprocess_text(text):
            cleaned_text = clean_text(text)
            sequence = tokenizer.texts_to_sequences([cleaned_text])
            # Pad sequence to the max length used during training
            padded = pad_sequences(sequence, maxlen=X_train_tf_reshaped.shape[2])
            # Reshape to include the time step dimension
            return np.reshape(padded, (padded.shape[0], 1, padded.shape[1]))
```

Code for UI based on the Best Model-Cont-1....

```
@app.route('/')
def index():
    return render_template_string('''
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Chatbot Interface</title>
        <style>
            body {
                font-family: Arial, sans-serif;
                background-color: #f0f0f0;
                margin: 0;
                padding: 0;
                display: flex;
                justify-content: center;
                align-items: center;
                height: 100vh;
            }
            .container {
                background-color: #fff;
                padding: 20px 30px;
                border-radius: 10px;
                box-shadow: 0 0 10px rgba(0, 0, 0,0.1);
                width: 40%;
                max-width: 900px;
                min-width: 600px;
                text-align: center;
            }
            h1 {
              margin-bottom: 20px;
              color: #333;
            }
            label {
                display: block;
                margin-bottom: 10px;
                color: #555;
                font-size: 1.2em;
            }
            input[type="text"] {
                width: 100%;
                padding: 10px;
                margin-bottom: 20px;
                border: 1px solid #ccc;
                border-radius: 5px;
                font-size: 1em;
            }
            button {
```

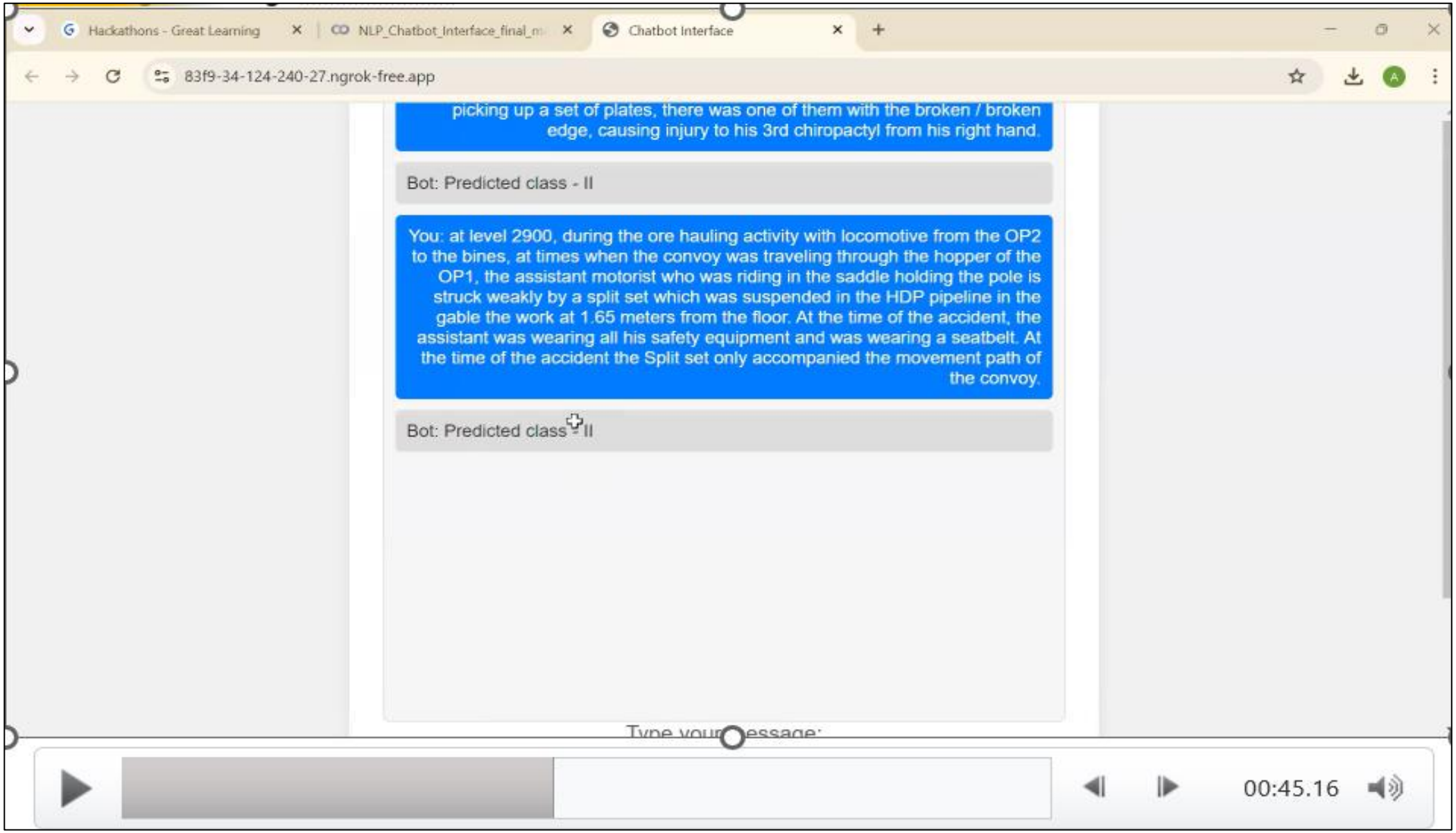
Code for UI based on the Best Model-Cont-2....

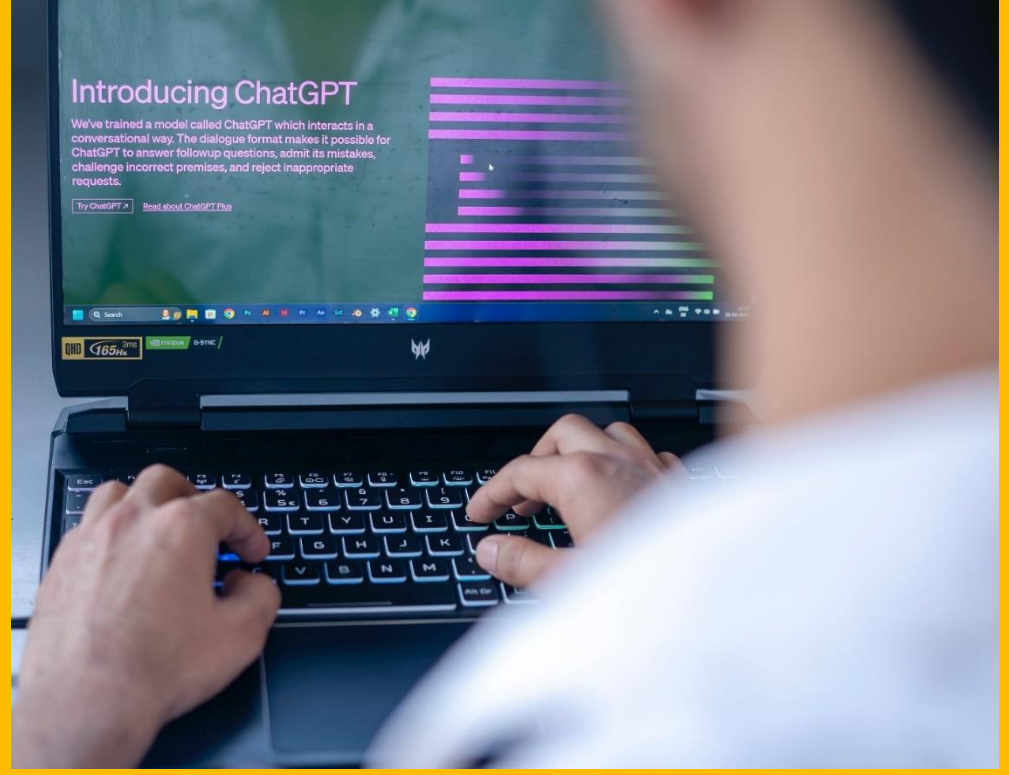
```
            background-color: #007bff;
            color: #fff;
            border: none;
            padding: 10px 20px;
            border-radius: 5px;
            font-size: 1em;
            cursor: pointer;
            transition: background-color 0.3s ease;
        }
        button:hover {
            background-color: #0056b3;
        }
        #chat-box {
            margin-top: 20px;
            padding: 10px;
            border: 1px solid #ccc;
            border-radius: 5px;
            background-color: #f5f5f5;
            height: 600px;
            overflow-y: auto;
            text-align: left;
        }
        .message {
            margin: 10px 0;
            padding: 10px;
            border-radius: 5px;
        }
        .user-message {
            background-color: #007BFF;
            color: #fff;
            text-align: right;
        }
        .bot-message {
            background-color: #ddd;
            color: #333;
            text-align: left;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Chatbot Interface</h1>
        <div id="chat-box"></div>
        <form id="chat-form">
            <label for="message">Type your message:</label>
            <input type="text" id="message" name="message" required>
            <button type="submit">Send</button>
```

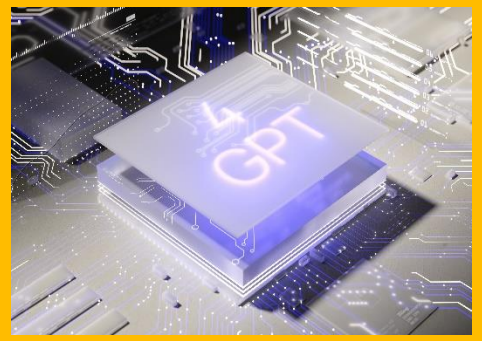
Code for UI based on the Best Model-Cont-3....

```
                userMessage.className = 'message user-message';
                userMessage.textContent = `You: ${message}`;
                chatBox.appendChild(userMessage);

                const responseDiv = document.createElement('div');
                responseDiv.className = 'message bot-message';
                responseDiv.textContent = 'Loading...';
                chatBox.appendChild(responseDiv);

                const response = await fetch('/predict', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/x-www-form-urlencoded',
                    },
                    body: `message=${encodeURIComponent(message)}`
                });

                const data = await response.json();
                responseDiv.textContent = `Bot: Predicted class - ${data.class}`;
                document.getElementById('message').value = '';

                // Scroll to the bottom of the chat box
                chatBox.scrollTop = chatBox.scrollHeight;
            });
        </script>
    </body>
    </html>
    ''')

@app.route('/predict', methods=['POST'])
def predict():
    message = request.form['message']
    processed_message = preprocess_text(message)
    prediction = model.predict(processed_message)
    predicted_class = prediction.argmax(axis=1)[0]

    class_to_roman = {0: 'I', 1: 'II', 2: 'III', 3: 'IV', 4: 'V'}
    roman_class = class_to_roman.get(predicted_class, "Unknown")

    return jsonify({'class': str(roman_class)})

if __name__ == '__main__':
    public_url = ngrok.connect(5000)
    print("Public URL:", public_url)
    app.run()
```

Industry Safety ChatBot UI Demo

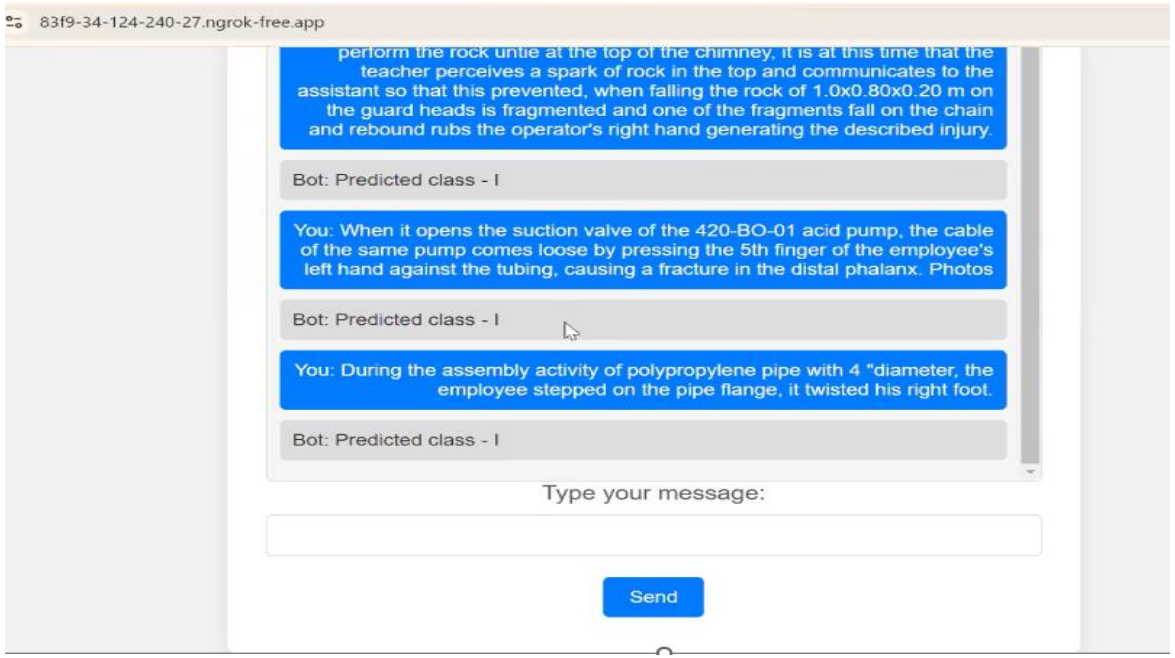






Milestone-3 Conclusion

- Further to Milestone-2 we have pickled BiGRU model for building the UI
- We used pyngrok, Flask, request, jsonify, render_template_string packages to build the UI
- As a result we were able to successfully deploy the UI
- It has been tested by passing two set of descriptions and the model was able to predict the Potential accident level severity accurately.
- A detail video representation has been added as a slide to give more details on the UI and screen reference under




THANK YOU