Natural Language Processing

Industrial Risk Analysis

A Capstone Project

Submitted by:

Subhra Ojha

Sid

Nagapraveen

Niranjan

Richa

In fulfilment of the requirements for the award of

**Post Graduate Program in**

**Artificial Intelligence & Machine Learning**

from:

**Great Learning**

**&**

**TEXAS McCombs**

The University of Texas at Austin

**Mentor:**

Aniket

March 2024

Table of Contents

|  |  |  |
| --- | --- | --- |
| SL NO | Topics | Page No |
| 1 | Introduction |  |
| 2 | Problem statement |  |
| 3 | Exploring input data |  |
| 3.1 | Univariate analysis |  |
| 3.2 | Bivariate analysis |  |
| 4 | Data preprocessing |  |
| 5 | Data Preparation for the Model Building |  |
| 6 | Model Building |  |
| 6.1 | Input Data Options |  |
| 6.2 | Model Training |  |
| 6.3 | Model Evaluation |  |
| 6.4 | Model Improvement |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. Introduction

Natural language processing (NLP) is a machine learning technology that gives computers the ability to interpret, manipulate, and comprehend human language.

Organizations today have large volumes of voice and text data from various communication channels like emails, text messages, social media newsfeeds, video, audio, and more.

They use NLP software to automatically process this data, analyze the intent or sentiment in the message, and respond in real time to human communication.

1. Problem Statement

The problem statement dataset is from Industrial safety domain. Now a days, it is an urgent need for industries/companies around the globe to understand why employees still suffer some injuries/accidents in plants or industries. Sometimes they also die in such environment. It is really crucial to analyse different feedbacks of employees and provide adequate measures to mitigate such challenges.

NLP (Natural Language Processing) is a great tool to analyse reviews of the employees and come up with detrimental measures to resolve the problem.

The database comes from one of the biggest industry in Brazil and in the world. 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.

The dataset has 425 records with 11 columns. There are 8 categorical fields, 1 description or text type and 2 fields as sequence and date. “Unnamed 0” is a sequence and “Data” is a date field ,hence they do not provide any adequate information for the analysis. These can be dropped for the analysis.

Text field “Description” is used for NLP analysis and other 6 categorical fields such as (**Countries, Local, Industry Sector, Genre, “Employee or Third Party”, “Critical Risk”**) are used as input data set. “**Potential Accident Level**” is considered as target variable. There is an equivalent target variable “**Accident Level**” which is not considered as target.

The aim of the project is to predict Potential accident level as per Description and other 6 categorical fields.

As part of the project, different ML techniques as **logistic regression, random forest, KN classifier, Decision Tree, adaptive boosting, gradient adaptive boosting, XG boosting** are used to determine accuracy. Different tuning techniques **as cross validation, Kfold, Stratified Kfold, grid search** are used to improve performance.

**Tensorflow, nltk, sklearn, matplotlib** are the major python libraries for the project.

1. Exploring the Input data

There are 425 records and 11 fields in the dataset. There are 14 duplicates and it leaves 411 final records to work on.

**3.1 Univariate Analysis-**

A function is created to perform univariate analysis.

def univariate\_analysis(dataframe, column, normalize\_data = False):

# Getting the number of unique values

print(f"The column '{column}' has {dataframe[column].nunique()} unique values.\n")

print(f"Unique values are:\n")

print(f"{dataframe[column].unique()}\n")

print(f"The value counts of data in this column are:\n{dataframe[column].value\_counts(normalize = normalize\_data)}")

# Plotting graphs

if dataframe[column].dtype == 'object':

fig, ax = plt.subplots(nrows = 1, ncols = 1, figsize = (8,6))

ax.set\_title(f"Countplot of data in column {column}.", fontsize = 12, pad = 15)

sns.countplot(data = dataframe, x = column, ax = ax)

ax.set\_xlabel(f"Column: {column}", fontsize = 12, labelpad = 12)

ax.set\_ylabel(f"Count", fontsize = 12, labelpad = 12)

if len(dataframe[column].value\_counts()) > 3:

ax.set\_xticklabels(ax.get\_xticklabels(), rotation = 90)

plt.show()

elif dataframe[column].dtype == 'int' or dataframe[column].dtype == 'float':

fig, ax = plt.subplots(nrows = 1, ncols = 1, figsize = (8,6))

ax.set\_title(f"Histogram of data in column {column}.", fontsize = 12, pad = 15)

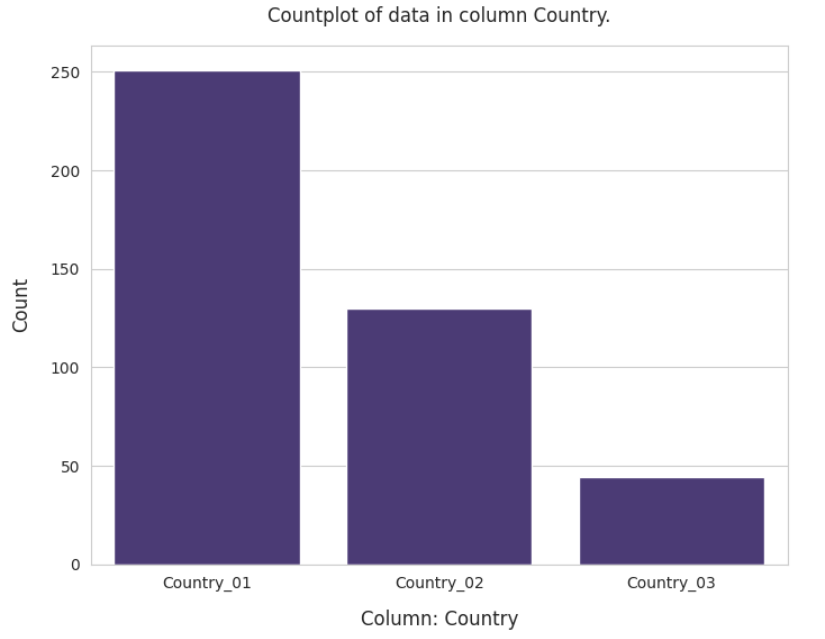
sns.histplot(data = dataframe, x = column, kde = True, ax = ax)

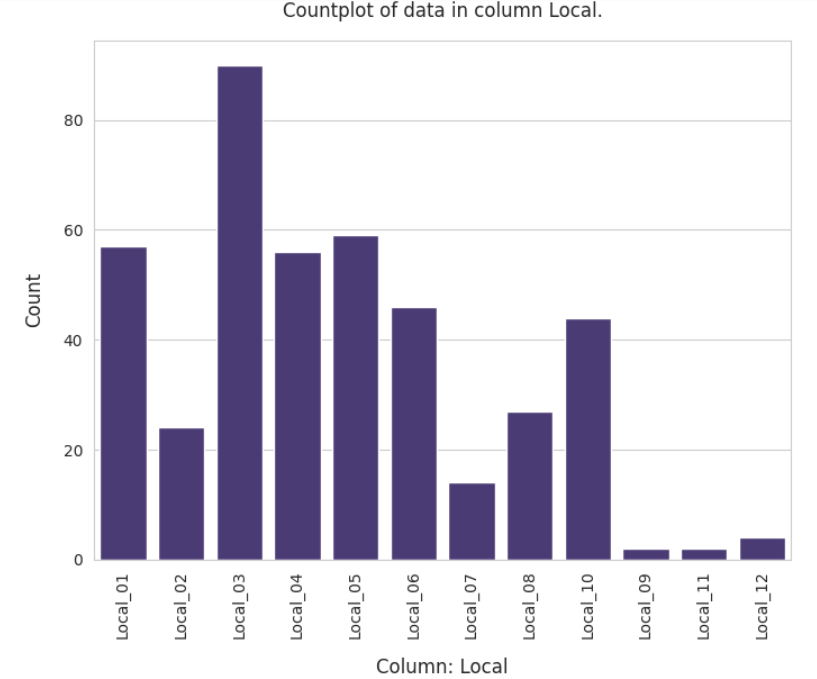
ax.set\_xlabel(f"Column: {column}", fontsize = 12, labelpad = 12)

ax.set\_ylabel(f"Frequency", fontsize = 12, labelpad = 12)

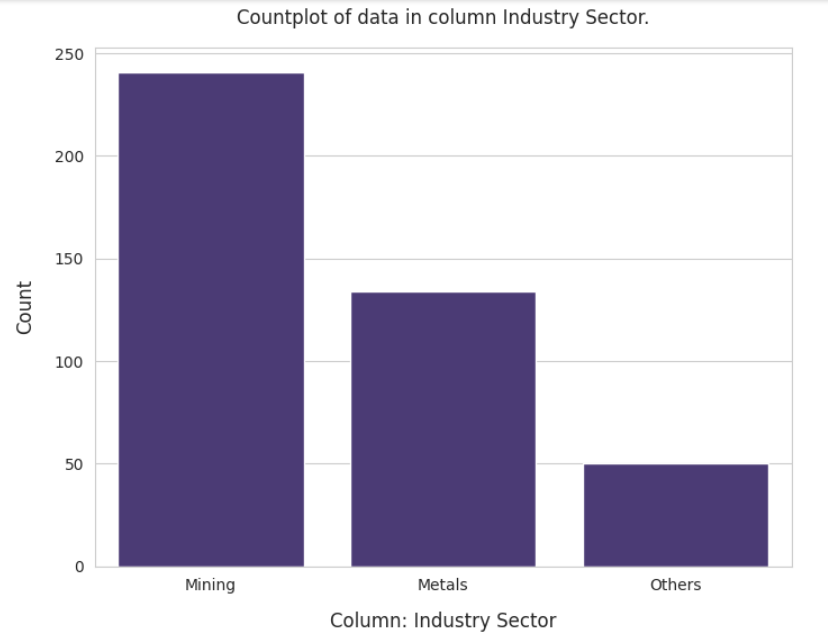
plt.show()

Outlining below univariate analysis of all categorical fields.

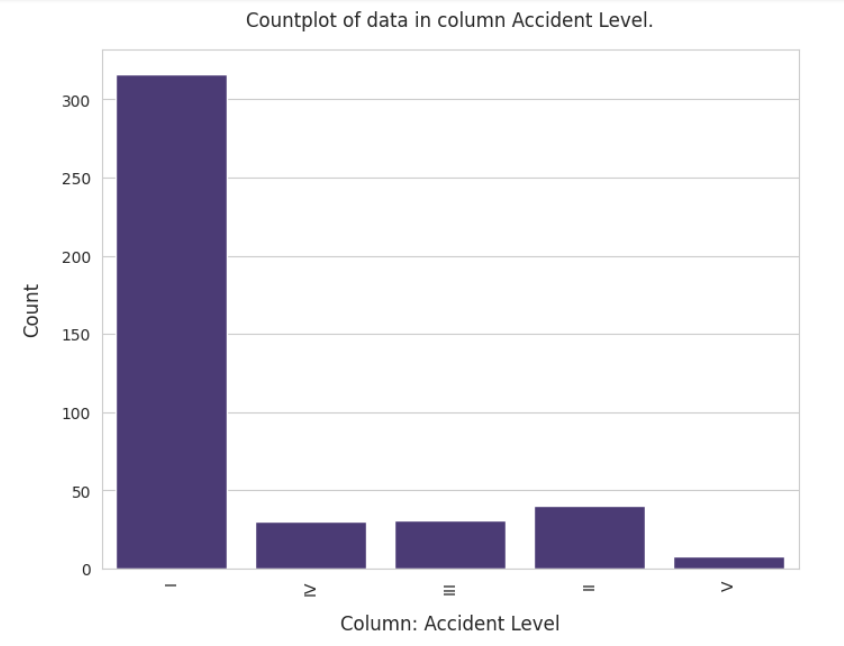
1. Country-
2. 251 out of the 425 accidents (approx. 59%) occured in Country\_01, followed by Country\_02 (approx. 30.6%) and Country\_03
3. Local



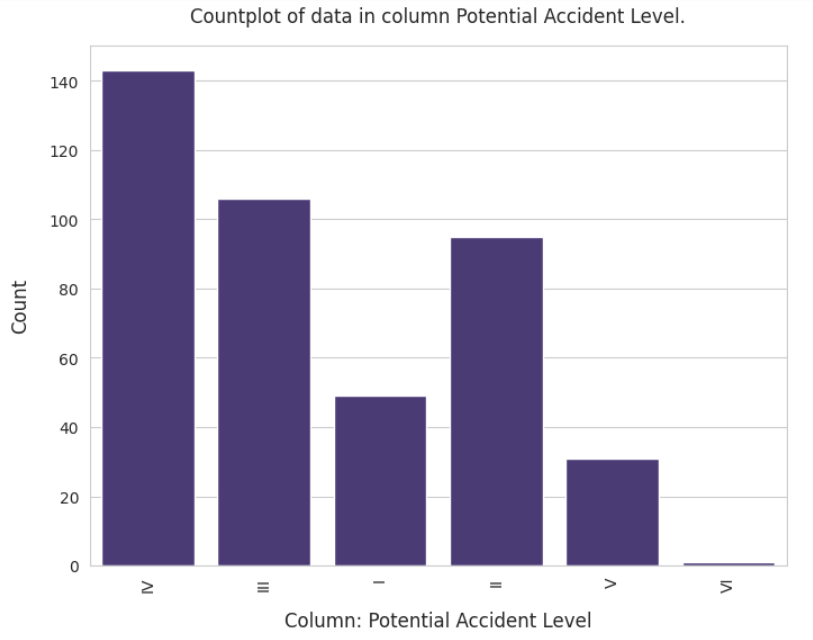
1. City 3 has the highest concentration of manufacturing plants in this dataset
2. City 9 has the lowest.
3. Cities 9, 11 and 12 have the lowest clustering of manufacturing plants, while Cities 1, 4, 5, 6 and 10 have relatively higher concentrations.
4. Industry Sector



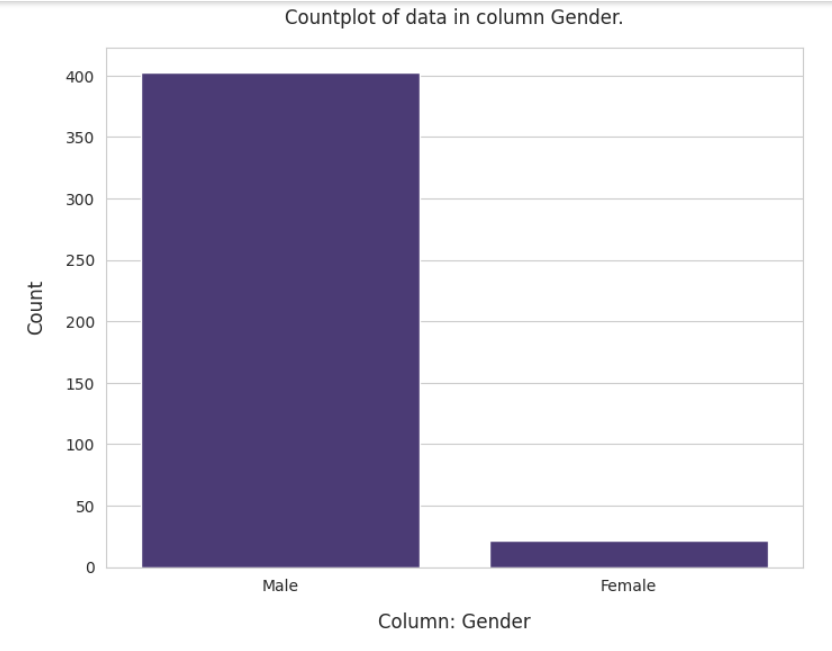
1. Most of the manufacturing plants are specific to the mining sector (approx. 56.7%), followed by Metals (approx. 31.5%)
2. Accident Level



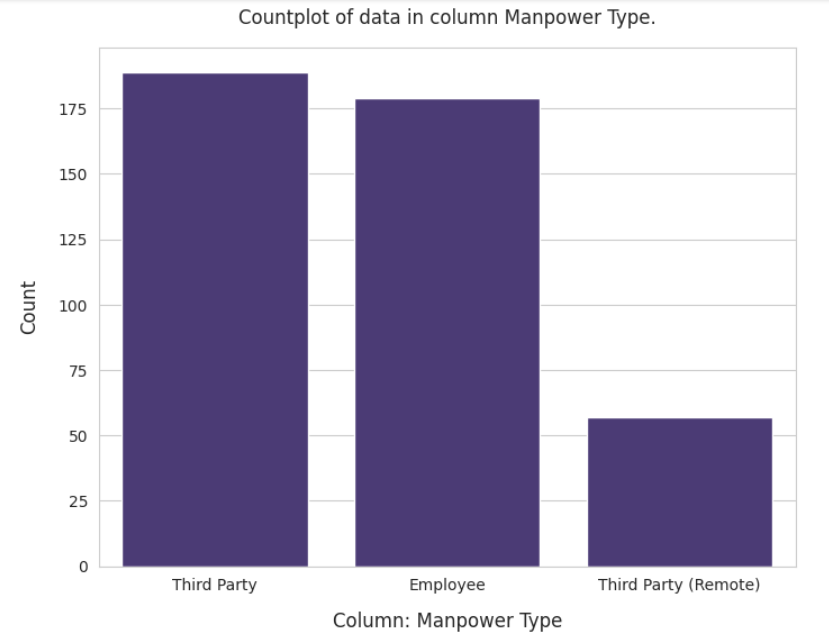
1. The target variable is highly skewed, with approx. 75% of the accidents being classified as Accident Level I
2. **We will not consider this as our primary target variable**
3. Potential Accident Level



1. Unlike the 'Accident Level' column, here Level IV has the highest representation (34%), followed by Level III (25%), Level II (22%), Level I (12%) and Level V (7%).
2. Also unlike the 'Accident Level' column, there is an extra Potential Accident level - Level VI - in this column. However, there is only 1 instance of this Potential Accident Level.
3. Level VI has only 1 record and it is merged with level V.
4. Gender



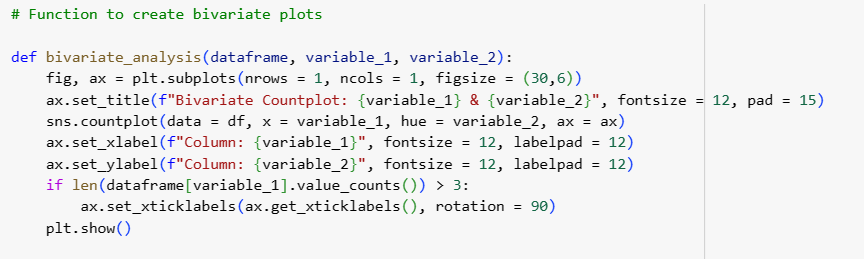
1. Representation of Male is imbalanced this feature, representing 95% of the total values.
2. Manpower Type



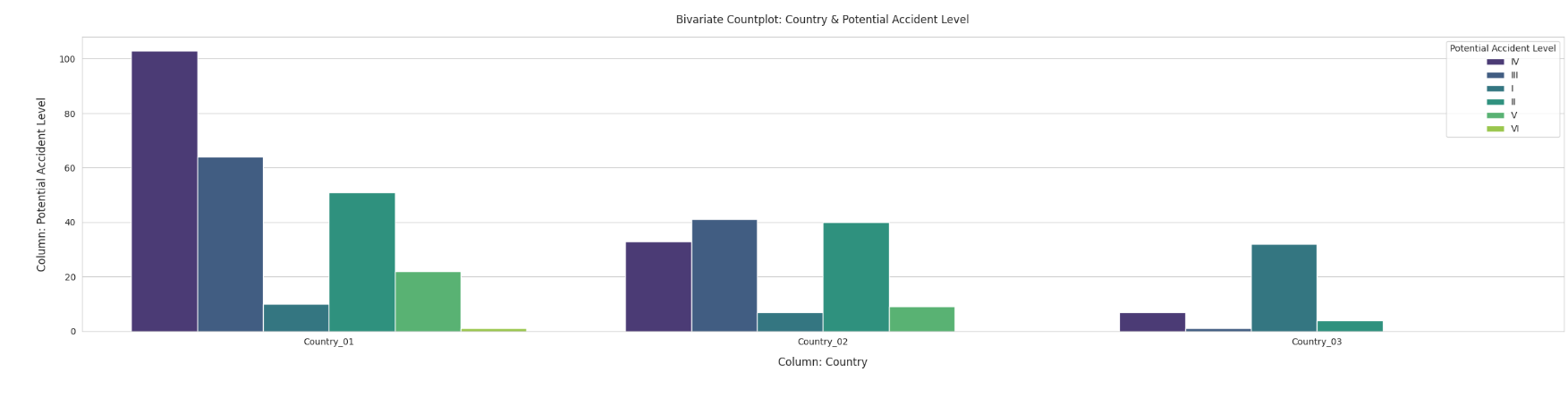
1. Both **Third Party and in-house** employees have a balanced representation.

**3.2 BIVARIATE PLOTS - AGAINST TARGET VARIABLE**

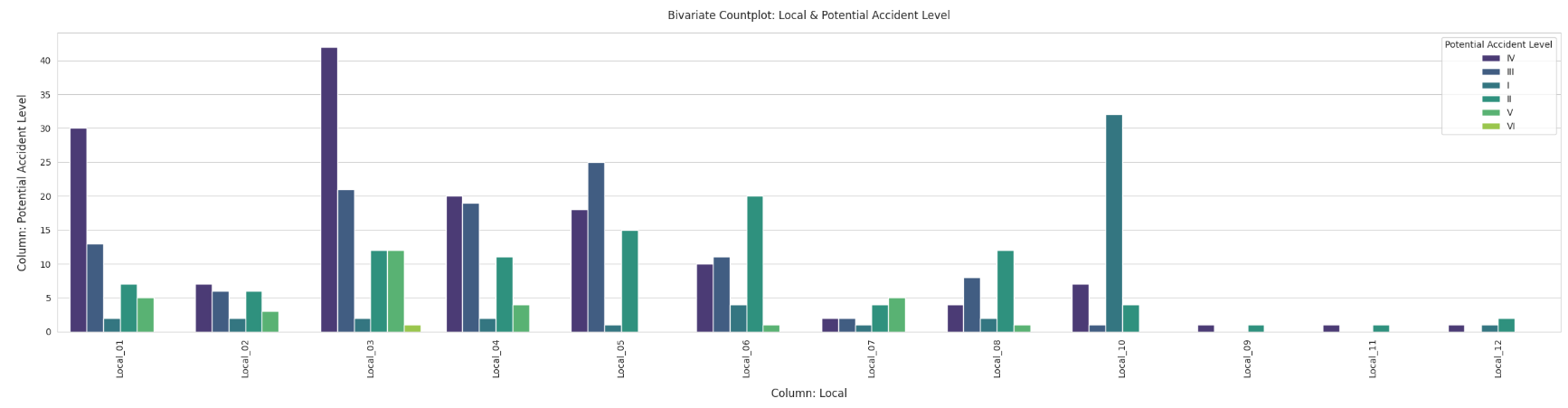
**This module shows bivariate analysis between target variable and feature columns. A function is created to perform it.**

****

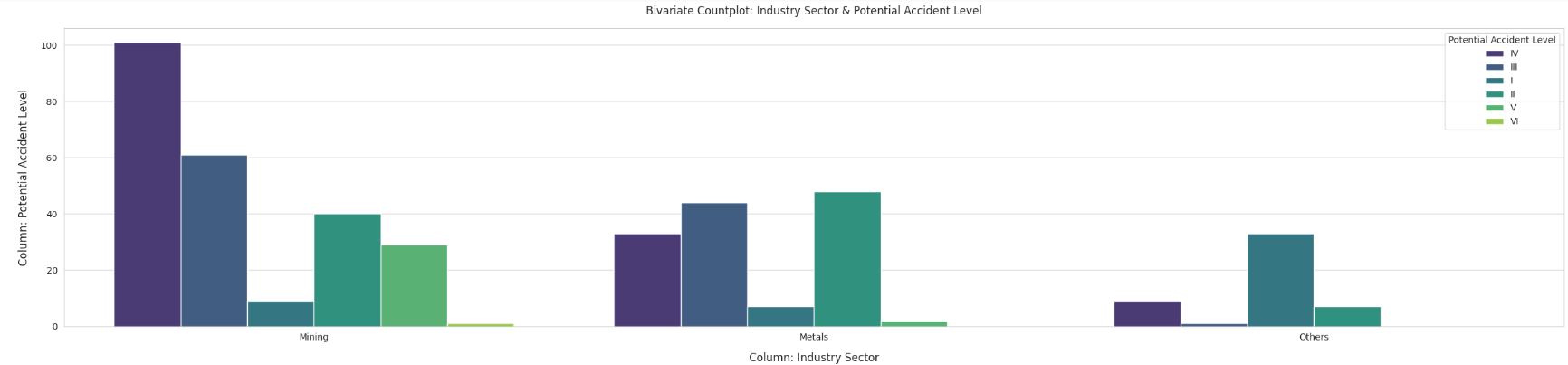
1. Country and Potential Accident Level



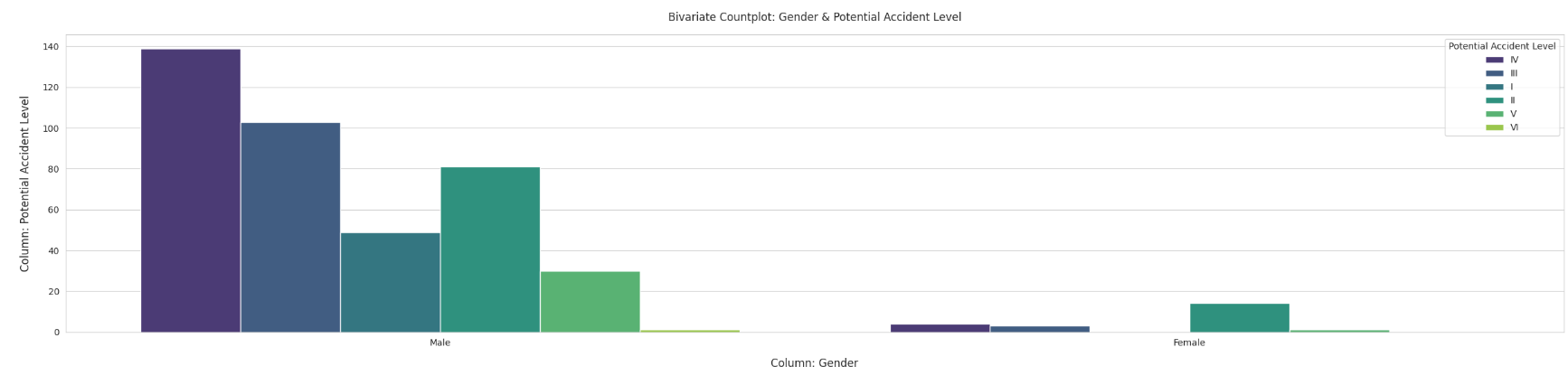
1. Local and Potential Accident Level



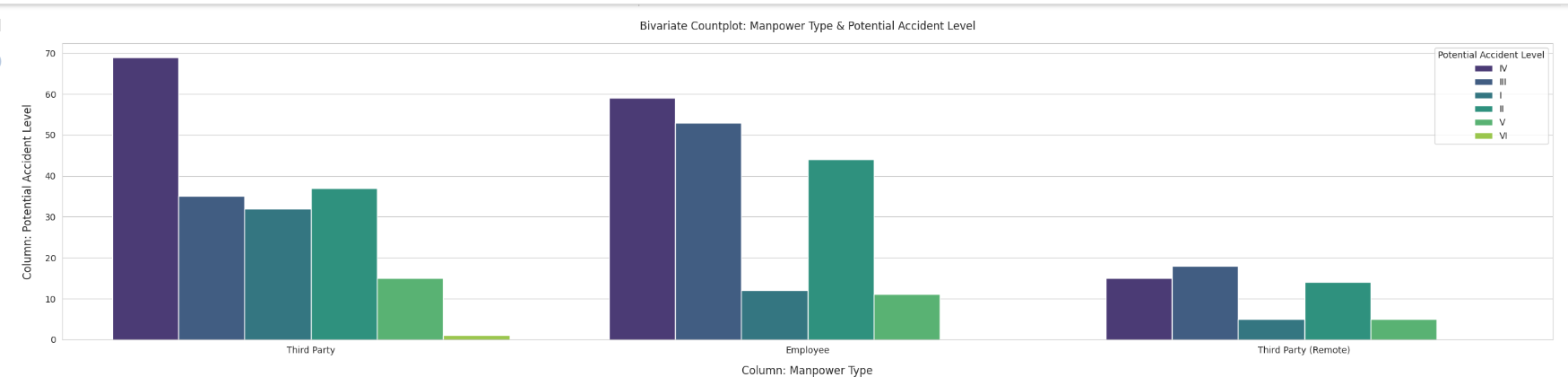
1. Industry sector and Potential Accident Level



1. Gender and Potential Accident Level



1. Manpower type and Potential Accident Level



Placeholder for Description

Placeholder for multivariate

1. Data Preprocessing

**‘Description’ field needs data cleansing as part of preprocessing. Then this field needs to be converted to numbers to fit to any model.**

**Below are cleansing techniques applied.**

1. Cleaning URL & HTML tags, if any
2. Removing any element that is not a word or whitespace character
3. Converting text to lowercase
4. Expanding contracted words like 'isn't' or 'wouldn't'
5. Removing stopwords, except for the word '**not**', since this word would provide context to sentences
6. Cleaning unncessary whitespaces around words
7. Removing numbers
8. Lemmatizing the words

Placeholder for pre and post formatting of description field.

Placeholder for wordcloud

Placeholder for count bar for 10 most frequent words

1. Data Preparation

**Description field needs to be prepared to fit to any ML or NN model. Below are some methods to be used for preparation.**

1. Label Encode 'Potential Accident Level'
2. Stopwords tweaking

a. **Group to go through df\_text and compare raw descriptions with cleaned descriptions to understand if certain keywords have gotten removed in the cleaning process**

b. Include words - 'while', 'when', 'during' .. **add to this list** - Done

c. Remove words like 'am', 'pm', units of measurement like 'kg', 'gram' etc. - Retained since they give context

d. Check if any commonly occuring, relevant technical words need to be expanded

1. Decide on the total length of most frequent words to be used.
2. Tokenization
3. Padding
4. Vectorization/Embedding-
   1. Count Vectorizer
   2. TFIDF
   3. Word2vec

Placeholder for Code snippet

Placeholder for POS tagging visualization

1. Model Building

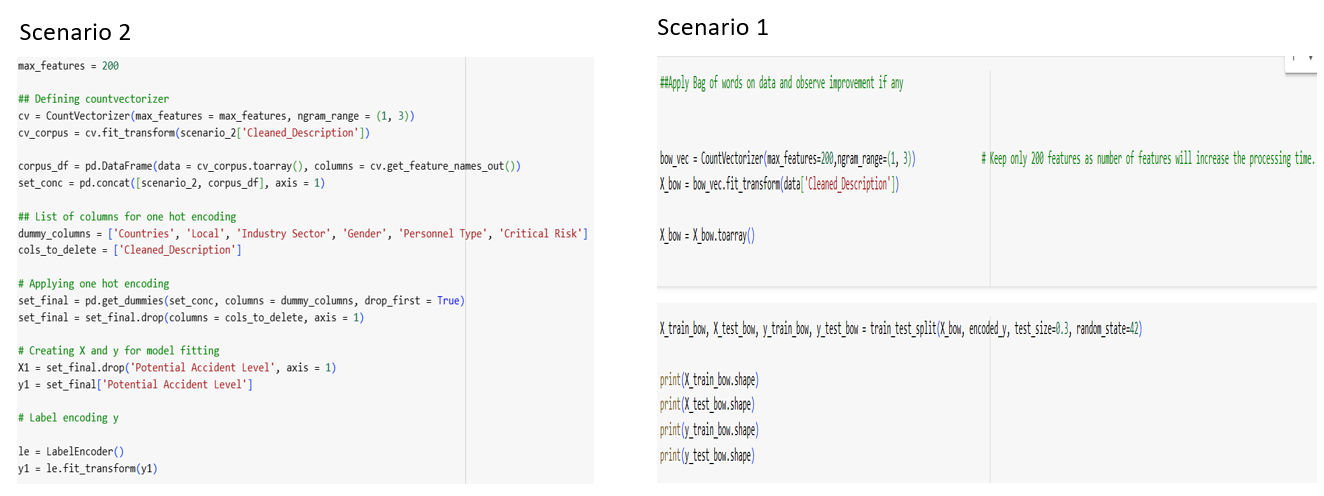
Dataset Preparation-

The dataset is divided into 2 types. Different models are tested against both scenarios.

**Scenario 1-** "Cleaned Description" will be feature and "Potential Accident Level" will be target Variable.

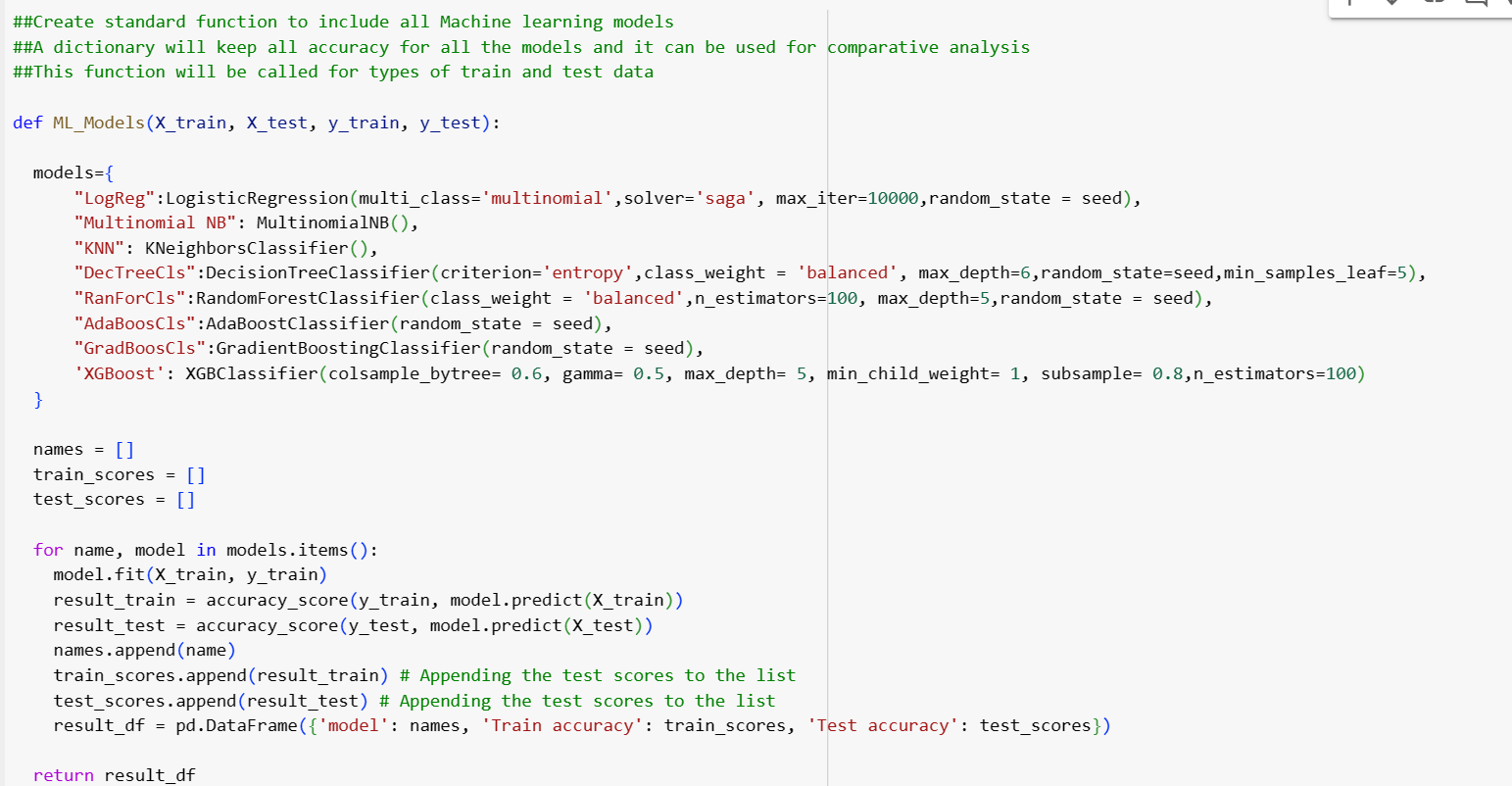
**Scenario 2-** All columns except "Data", "Accident Level" will be features. "Potential Accident Level" will be target Variable. All columns will be one hot encoded and Cleaned Description is passed through count vectorizer. Target variable “Potential Accident Level” will be Lebel-encoded.

*Note- An assumption is considered to include Potential Accident Level VI to V as there is only 1 record present in level VI.*

****

Model Building-

A function is created to combine all models and it performs model fitting and predict accuracy of both train and test model. The function takes input as train and test dataset of X and Y.



A number of models are used as below. Set of parameters are considered for each of the models.

*A.Logistic Regression-*

i. Multi\_class = ‘multinomial’ : It is considered as the target dataset is multiclass problem.

ii. Solver= ‘saga’ : For multiclass problems, only ‘newton-cg’, ‘sag’, ‘saga’ and ‘lbfgs’ handle multinomial loss.

iii. Max\_iter= 10000: It gave a better result in accuracy. Experimented with different options.

*B. Multinomial NaiveBayes*

*C. Kneighbour Classifier*

*D. Decision Tree Classifier-*

i. Criterion = ‘entropy’ : Entropy represents order of randomness. It is performing better compared to Gini.

ii. class\_weight= ‘balanced’ : The “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n\_samples / (n\_classes \* np.bincount(y))

iii. max\_depth=6 : The maximum depth of the tree.  Best value is 6 with all experiments.

iv. min\_samples\_leaf=5: The minimum number of samples required to be at a leaf node. Best value is 5 with all experiments.

*E. Random Forest Classifiers-*

i. n\_estimators=100: The n\_estimators parameter specifies the number of trees in the forest of the model. Best value is 100 with all experiments.

*F. Ada Boosting Classifiers*

*G. Gradient Boosting Classifiers*

*H. XGBoosting Classifiers-*

i. colsample\_bytree= 0.6 : It is the subsample ratio of columns when constructing each tree.

* ii. gamma= 0.5: Minimum loss reduction required to make a further partition on a leaf node of the tree. The larger gamma is, the more conservative the algorithm will be.

iii. min\_child\_weight= 1: Minimum sum of instance weight (hessian) needed in a child.

iv. subsample= 0.8: Subsample ratio of the training instances.

Model training-

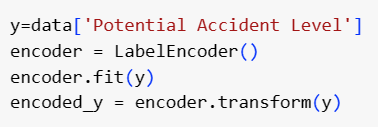
The **feature variable (X)** is transformed with 4 methods. We will test the models with each of the method and compare the accuracy to find the best method. 

Maxlen- It will limit the total sequence returned so that it has a maximum length. 200 is considered as max length of any line is 659 and minimum being 61. So a length is considered in between.

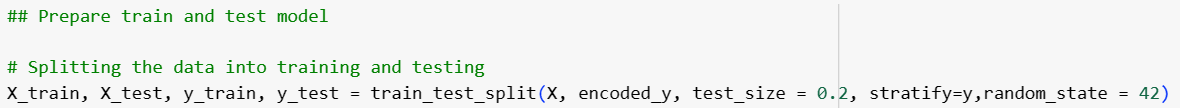
Max\_features- The maximum number of words to keep, based on word frequency.

Oov\_taken: It is out of vocabulary words and its count.

**Target variable (Y)** is transformed with Levelencoder method. Level encoder justified Y as it is ordinal data in the dataset.

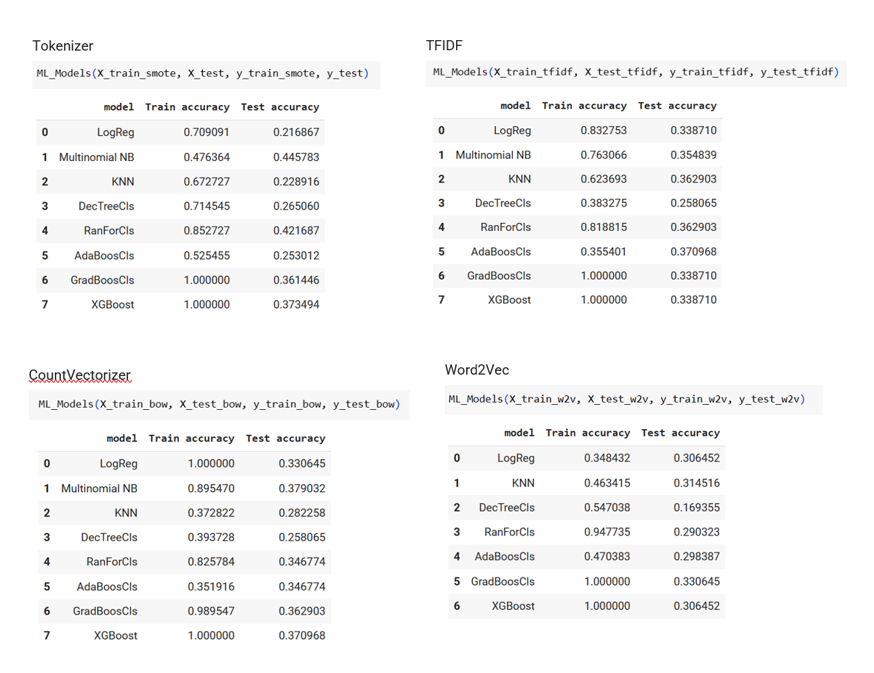


Model is split into train and test after feature variable transforming with any of above method.

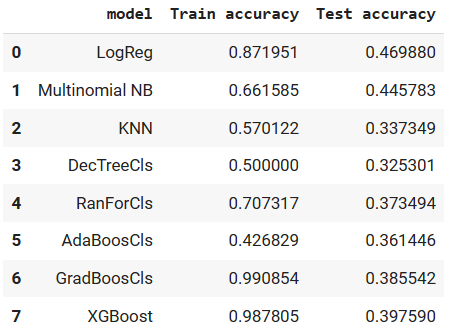


Model Result-

Below diagram depicts a comparative analysis of models accuracy results.



**Scenario 2**



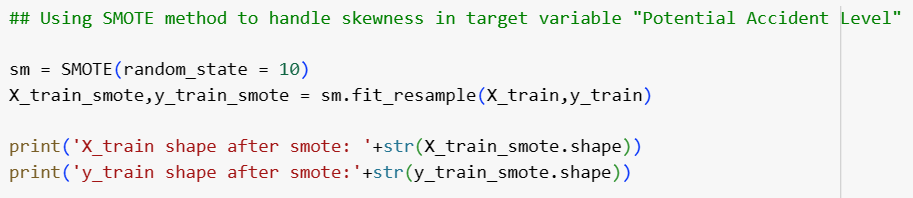
Clearly **Multinomial NB** performs best among all ML models with tokenizer method. The accuracy is around 45%. It is definitely not the best result in terms of accuracy. So it concludes that all ML models are not best suited for the prediction.

Even when we consider scenario 2, Multinomial NB is giving approx. 45% of accuracy. There is considerable amount of difference of accuracy between train and test. There is not much visible improvement in accuracy with inclusion of all columns.

Model Tuning-

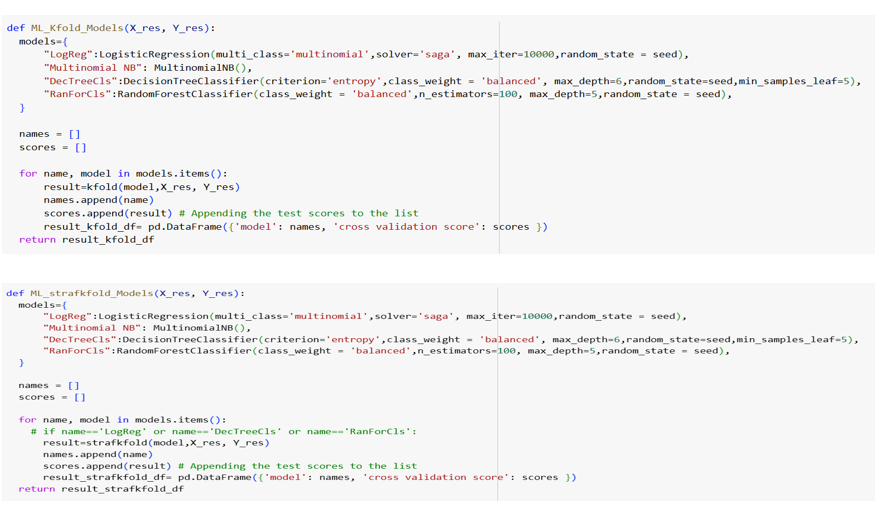
There are different tuning methods which can used to improve the accuracy.

* SMOTE or SMOTETomek methods to handle class imbalance in target variable



* Cross validation methods

Functions are created for Kfold and Stratified Kfold methods. X and Y are input to the functions. Result shows cross validation score of all the models to give a comparative analysis.



* Gridsearch CV methods

A list of hyper-parameters are used for gridsearchCV method. It provided best hyper-parameter which will give best score.



**Best parameters are as follows-**

*a. RandomForestClassifier*

{'ccp\_alpha': 0.001, 'criterion': 'entropy', 'max\_depth': 12, 'max\_features': 'log2'}

*b. AdaBoostClassifier*

{'learning\_rate': 0.1, 'n\_estimators': 500}

*c. GradientBoostingClassifier*

{'learning\_rate': 0.05, 'max\_depth': 3, 'n\_estimators': 1000}

*d. XGBClassifier*

{'gamma': 0.5, 'max\_depth': 3, 'n\_estimators': 50}

*e. LogisticRegression*

{'C': 100, 'penalty': 'l2'}

*f. KNeighborsClassifier*

{'metric': 'manhattan', 'n\_neighbors': 1, 'weights': 'uniform'}

*g. Multinomial NB*

{'alpha': 0.5, 'fit\_prior': True}

A function is created to find accuracy of different models with best found hyper parameters.



Model Performance-

We have considered MultinomialNB for the detailed analysis.

