*A project report on*

# Construction site accident analysis using text mining and natural language processing techniques

*Submitted in partial fulfillment for the award of the degree of*

**M.Tech (Software Engineering)**

*by*

**SUDHARSHAN P (16MIS0100)**

**SITE**

November, 2020

## DECLARATION

I hereby declare that the thesis entitled "Construction site accident analysis using text mining and natural language processing techniques" submitted by me, for the award of the degree of M.Tech (Software Engineering) VIT is a record of bonafide work carried out by me under the supervision of Prof.Nirmala.M

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

**Place: Vellore** **Sudharshan P**

**Date:** **Signature of the Candidate**

## CERTIFICATE

This is to certify that the thesis entitled “Construction site accident analysis using text mining and natural language processing techniques**”** submitted by Sudharshan P (16MIS0100) SITE VIT, for the award of the degree of M.Tech(Software Engineering)is a record of bonafide work carried out by her under my supervision.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The Project report fulfils the requirements and regulations of VIT and in my opinion meets the necessary standards for submission.

[illegible]

**Signature of the Guide** **Signature of the HOD**

**Internal Examiner** **External Examiner**

## ABSTRACT

Workplace safety is a major concern in many countries. Among various industries, construction sector is identified as the most hazardous work place. Construction accidents not only cause human sufferings but also result in huge financial loss. To prevent reoccurrence of similar accidents in the future and make scientific risk control plans, analysis of accidents is essential. In construction industry, fatality and catastrophe investigation summary reports are available for the past accidents. In this study, text mining and natural language process (NLP) techniques are applied to analyse the construction accident reports. To be more specific, five baseline models, support vector machine (SVM), linear regression (LR), K-nearest neighbor (KNN), decision tree (DT), Naive Bayes (NB) and an ensemble model are proposed to classify the causes of the accidents. Besides, logistic regression algorithm is utilized to optimize weight of each classifier involved in the ensemble model. Experiment results show that the optimized ensemble model outperforms rest models considered in this study in terms of average weighted F1 score. The result also shows that the proposed approach is more robust to cases of low support. Moreover, an unsupervised chunking approach is proposed to extract common objects which cause the accidents based on grammar rules identified in the reports. As harmful objects are one of the major factors leading to construction accidents, identifying such objects is extremely helpful to mitigate potential risks. Certain limitations of the proposed methods are discussed and suggestions and future improvements are provided

## ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Prof.Nirmala.M, Assistant Professor, School of Information Technology and Engineering, Vellore Institute of Technology, for his constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavor. My association with him / her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Machine Learning.

I would like to express my gratitude to Dr.G.Viswanathan, Shri.Sekar Viswanathan, Dr.Anand A.Samuel, Dr.S.Narayanan, and Balakrushna Tripathy, School of Information Technology and Engineering, for providing with an environment to work in and for his inspiration during the tenure of the course.

In jubilant mood I express ingeniously my whole-hearted thanks to Dr.S.Sree Dharinya, Head of the Department all teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: Vellore

Date: Name of the student: Sudharshan.P

# TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES:

## LIST OF ABBREVIATIONS

| | |
|---|---|
| KNN | K-nearest neighbor |
| SVM | Support vector machine |
| NLP | Natural language process |
| NB | Naive Bayes |
| LR | Linear regression |
| TF-IDF | Term and inverse document frequency |
| LRM | Logistic regression algorithm |

# CHAPTER 1

# INTRODUCTION

## 1. Introduction

Past accident data contains details of accidents and by building machine learning algorithms model we can analyse data to identify cause of accident and can prevent future accident by giving test data of new work to predict causes of accidents and can avoid such causes. This machine learning algorithms can help in extracting dangerous objects such as misused tools, sharp objects nearby, damaged equipment etc. In this project is describing concept to provide safety to workers at construction site from accidents by analysing past accident data by using machine learning algorithms and text mining technique such as TF-IDF (Term Frequency-Inverse Document Frequency) and natural language text processing to remove special symbols, stop words, stemming etc. various texting mining and NLP techniques are explored with respect to construction site accidents analysis. Using this technique we will remove stop words, punctuations, special symbols and apply stemming technique to clean past accident data. After data cleaning we will convert all text data to numeric vector by using TF-IDF technique. TF-IDF contains frequency weight of each word in vector and using this vector we will build machine learning train model. Whenever we give new test data then that test data also convert to TF-IDF and then apply on train model to search for similar data and give output of similar data as prediction

## 1.1 OBJECTIVE OF THE PROJECT

The framework consists of three classification models used to classify date fruit images in real time according to their type, maturity, and harvesting decision. In the classification models, deep convolutional neural networks are utilized with transfer learning and fine-tuning on pre-trained models. To build a robust vision system, we create a rich image dataset of date fruit bunches in an orchard that consists of more than 8000 images of five date types in different pre-maturity and maturity stages.

## 1.2 EXISTING SYSTEM

Past accident data contains details of accidents and by building machine learning algorithms model we can analyse data to identify cause of accident and can prevent future accident by giving test data of new work to predict causes of accidents and can avoid such causes. This machine learning algorithms can help in extracting dangerous objects such as misused tools, sharp objects nearby, damaged equipment etc.

## 1.3. PROPOSED SYSTEM

This project is describing concept to provide safety to workers at construction site from accidents by analysing past accident data by using machine learning algorithms and text mining technique such as TF-IDF (Term Frequency-Inverse Document Frequency) and natural language text processing to remove special symbols, stop words, stemming etc.

## 1.4 LITERATURE SURVEY

### TITLE: 1 DEVELOPMENT AND EVALUATION OF A NAÏVE BAYESIAN MODEL FOR CODING CAUSATION OF WORKERS' COMPENSATION CLAIMS

**AUTHORS:** S.J. Bertke, A.R. Meyers, S.J. Wurzelbacher, J. Bell, M.L. Lampl, D. Robins

**YEAR:** 2012

**DESCRIPTION:**

Development and evaluation of a Naïve Bayesian model for coding causation of workers' compensation claims developed a Naïve Bayesian model to classify the compensation claims causation due to work related injuries. The proposed model achieved an overall accuracy of approximately 90%, however the accuracy of claims belongs to minor injury categories dropped.

## TITLE:2 NEAR-MISS NARRATIVES FROM THE FIRE SERVICE: A BAYESIAN ANALYSIS

**AUTHORS:** J.A. Taylor, A.V. Lacovara, G.S. Smith, R. Pandian, M. Lehto,

**YEAR:** 2014

**DESCRIPTION:**

Near-miss narratives from the fire service: A Bayesian analysis  applied Naïve Bayesian and Fuzzy models to categorize the injury outcome and mechanism of injury for fire service incident reports extracted form from the National Firefighter Near-Miss Reporting System. Results showed that Fuzzy model achieved a sensitivity of 0.74 while sensitivity of Naïve Bayesian model is 0.678.

## TITLE: 3 COMPUTERIZED CODING OF INJURY NARRATIVE DATA FROM THE NATIONAL HEALTH INTERVIEW SURVEY

**AUTHORS:** H.M. Wellman, M.R. Lehto, G.S. Sorock, G.S. Smith,

**YEAR:** 2004

**DESCRIPTION:**

Computerized coding of injury narrative data from the National Health Interview Survey proposed a Fuzzy Bayesian model to classify injury narratives into external-cause-of-injury and poisoning (E-code) categories. Data used in this study is the injury reports from US National Health Interview Survey (NHIS) during 1997 and 1998. The proposed model achieved an accuracy of 87.2%.

## TITLE:4 EXTRACTING RECURRENT SCENARIOS FROM NARRATIVE TEXTS USING A BAYESIAN NETWORK: APPLICATION TO SERIOUS OCCUPATIONAL ACCIDENTS WITH MOVEMENT DISTURBANCE

**AUTHORS:** F. Abdat, S. Leclercq, X. Cuny, C. Tissot

**YEAR:** 2014

**DESCRIPTION:**

Extracting recurrent scenarios from narrative texts using a Bayesian network: Application to serious occupational accidents with movement disturbance applied Bayesian network to extract recurrent serious Occupational Accident with Movement Disturbance (OAMD) scenarios from narrative texts. It is noted that data pre-processing of this approach is time consuming and expert knowledge is required.

## TITLE:5 CLASSIFYING INJURY NARRATIVES OF LARGE ADMINISTRATIVE DATABASES FOR SURVEILLANCE—A PRACTICAL APPROACH COMBINING MACHINE LEARNING ENSEMBLES AND HUMAN REVIEW

**AUTHORS:** H.R. Marucci-wellman, H.L. Corns, M.R. Lehto,

**YEAR:** 2017

**DESCRIPTION:**

Classifying injury narratives of large administrative databases for surveillance—A practical approach combining machine learning ensembles and human review proposed an approach which combined manual coded rules with machine learning algorithms for injury narratives classification. Results showed that using Logistic Regression (LR) and filtering out the bottom 30% of its predictions reviewed manually resulted an overall sensitivity of 0.89.

## TITLE:6 COMPARISON OF METHODS FOR AUTO-CODING CAUSATION OF INJURY NARRATIVES

**AUTHORS:** S.J. Bertke, A.R. Meyers, S.J. Wurzelbacher,

**YEAR:** 2016

**DESCRIPTION:**

Comparison of methods for auto-coding causation of injury narratives compared the methods of Naïve Bayesian and Regularized Logistic Regression for auto coding the causation of injury narratives. Dataset used was from Occupational Injury and Ill-ness Classification System (OIICS), and results showed that the logistic model achieved an overall accuracy of 71% for 2-digit OIICS event/ exposure classification system and 87% for first digit respectively. In terms of the analysis of construction related accidents.

## TITLE:7 APPLICATION OF MACHINE LEARNING TO CONSTRUCTION INJURY PREDICTION

**AUTHORS:** A.J. Tixier, M.R. Hallowell, B. Rajagopalan, D. Bowman,

**YEAR:** 2016

**DESCRIPTION:**

Application of machine learning to construction injury prediction applied Random Forest (RF) and Stochastic Gradient Tree Boosting (SGTB) algorithms to predict type of energy involved in the accident, injury type, body part affected, and injury severity using construction injury reports. Rank Probability Skill Score (PRSS) of the proposed methods ranked from 0.236 to 0.436.

## TITLE: 8 AUTOMATED CONTENT ANALYSIS FOR CONSTRUCTION SAFETY: A NATURAL LANGUAGE PROCESSING SYSTEM TO EXTRACT PRECURSORS AND OUTCOMES FROM UNSTRUCTURED INJURY REPORTS

**AUTHORS:** A.J. Tixier, M.R. Hallowell, B. Rajagopalan, D. Bowman,

**YEAR:** 2016

**DESCRIPTION:**

Automated content analysis for construction safety: A natural language processing system to extract precursors and outcomes from unstructured injury report proposed a NLP approach based on hand crafted rules and keywords dictionary to extract outcomes and precursors from unstructured injury reports and achieved a recall of 0.97 and precision of 0.95, however the proposed approach was not robust to unanticipated situations.

## TITLE:9 CONSTRUCTION ACCIDENT NARRATIVE CLASSIFICATION: AN EVALUATION OF TEXT MINING TECHNIQUES

**AUTHORS:** Y.M. Goh, C.U. Ubeynarayana,

**YEAR:** 2017

**DESCRIPTION:**

Construction accident narrative classification: An evaluation of text mining techniques applied support vector machine (SVM), linear regression (LR), random forest (RF), K-nearest neighbor (KNN), decision tree (DT) and Naive Bayes (NB) algorithms for construction accident narrative classification. Among which, SVM achieved a F1 score ranged from 0.45 and 0.92 and outperformed the other classifiers.

## TITLE: 10 AN ENSEMBLE APPROACH FOR CLASSIFICATION OF ACCIDENT NARRATIVES

**AUTHORS:** A. Chokor, H. Naganathan, W.K. Chong, M. El,

**YEAR:** 2016

**DESCRIPTION:**

An Ensemble Approach for Classification of Accident Narratives  The author further presented an ensemble approach for construction accident narrative classification.

## TITLE: 11 ANALYZING ARIZONA OSHA INJURY REPORTS USING UNSUPERVISED MACHINE LEARNING

**AUTHORS:** H. Fan, H. Li,

**YEAR:** 2013

**DESCRIPTION:**

Analyzing Arizona OSHA Injury Reports Using Unsupervised Machine Learning applied a K-means based approach to classify injury reports. Four clusters were identified and each cluster represented a type of accident. Identified accident types were ‘falls', ‘struck by objects’, ‘electrocutions’ and ‘trenches collapse’ respectively.

## TITLE: 12 RETRIEVING SIMILAR CASES FOR ALTERNATIVE DISPUTE RESOLUTION IN CONSTRUCTION ACCIDENTS USING TEXT MINING TECHNIQUES

**AUTHORS:** Y. Zou, A. Kiviniemi, S.W. Jones,

**YEAR:** 2008

**DESCRIPTION:**

Retrieving similar cases for alternative dispute resolution in construction accidents using text mining techniques compared the text mining approach with case-based reasoning approach for accidents documents retrieval. Result showed that the text mining approach is superior in terms of recall and precision

# CHAPTER 2

# SYSTEM SPECFICATION

## 2.1 SOFTWARE REQUIREMENTS:

Functional requirements for a secure cloud storage service are straightforward:

1. The service should be able to store the user’s data;
2. The data should be accessible through any devices connected to the Internet; the service should be capable to synchronize the user’s data between multiple devices (notebooks, smart phones, etc.);
3. The service should preserve all historical changes (versioning);
4. Data should be shareable with other users;
5. The service should support SSO; and The service should be interoperable with other cloud storage services, enabling data migration from one CSP to another.

- **Operating System:** Windows
- **Coding Language**: Python 3.7

**HARDWARE REQUIREMENTS:**

- **Processor** - Pentium –III
- **Speed** – 2.4 GHz
- **RAM** - 512 MB (min)
- **Hard Disk** - 20 GB
- **Floppy Drive** - 1.44 MB
- **Key Board** - Standard Keyboard
- **Monitor** – 15 VGA Colour

## 2.2 ALGORITHM DESCRIPTION

### 2.2.1 SUPPORT VECTOR MACHINE

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyper plane. In other words, given labeled training data (*supervised learning*), the algorithm outputs an optimal hyper plane which categorizes new examples.

The basics of Support Vector Machines and how it works are best understood with a simple example. Let’s imagine we have two categories: red and blue, and our data has two features: x and y. We want a classifier that, given a pair of (x,y)coordinates, outputs if it’s either red or blue. We plot our already labeled training data on a plane: A support vector machine takes these data points and outputs the hyper plane (which in two dimensions it’s simply a line) that best separates the categories. This line is the decision boundary: anything that falls to one side of it we will classify as blue, and anything that falls to the other as red. But, what exactly is the best hyper plane. For SVM, it’s the one that maximizes the margins from both categories. In other words: the hyper plane (remember it’s a line in this case) whose distance to the nearest element of each category is the largest.

```
candidateSV = {closest pair from opposite classes} while there are violating points do
Find a violator
candidateSV = candidateSV S violator
if any αp < 0 due to addition of c to S then
candidateSV = candidateSV \ p
repeat till all such points are pruned
end if
end while
```

SVM is generally used for text categorization. It can achieve good performance in high-dimensional feature space. An SVM algorithm represents the examples as points in space, mapped so that the examples of the different categories are separated by a clear margin as wide as possible. The basic idea is to find the hyperplane which is represented as the vector w which separates document vector in one class from the vectors in other class.

### 2.2.2 K-NEAREST NEIGHBOR (KNN):

KNN algorithm is widely used for pattern classification based on feature similarity. For a given unclassified sample point, it is classified by a majority vote of its neighbors, with the point being assigned to the class most common among its K nearest neighbors. KNN is a lazy algorithm. Unlike most statistical methods which elaborate a model from the information available in the historic data, KNN considers the training set as the model itself. Thus there is no explicit training phase for KNN algorithm and during the testing phase, all training data is needed due to the lack of generalization. A KNN algorithm is characterized by issues such as number of neighbors, adopted distance, etc.

### 2.2.3 DECISION TREE:

Decision tree is a hierarchical tree-based classifier. It is represented by a set of nodes, a directional graph that starts at the base with a single node and extends to many leaf nodes that represent the categories that the tree can classify. It classifies a given sample data by applying a series of rules to features of the sample data. Each rule is represented by a node and each internal node points to one child node for each possible outcome corresponding to the applied rule. Such process is repeated until the sample data is sorted into a class by following the path from root node to leaf node. The sample data is assigned to the class which is the leaf it reaches.

### 2.2.4 LOGISTIC REGRESSION:

Logistic regression is an algorithm used to describe relationship between a response variable and other explanatory variables. In most cases, the response variable is discrete and consists of more than one possible value. In terms of classification problems, logistic regression method can be applied to predict the probability of a given sample data being assigned to a certain category.

### 2.2.5 NAIVE BAYESIAN:

Naïve Bayesian classifier is a member of probabilistic classifiers family which has been studied extensively since the 1950s. It was introduced under a different name into the text retrieval community in the early 1960s. It is widely used in text categorization tasks such as documents classification and spam email detection.

### 2.2.6 SEQUENTIAL QUADRATIC PROGRAMMING:

SQP was first proposed by Wilson .The algorithm relies on a profound theoretical foundation and has been proved to be both global convergent and locally super linearly convergent. Due to such advantages, SQP is one of the most efficient algorithms for solving constrained nonlinear optimization problems.

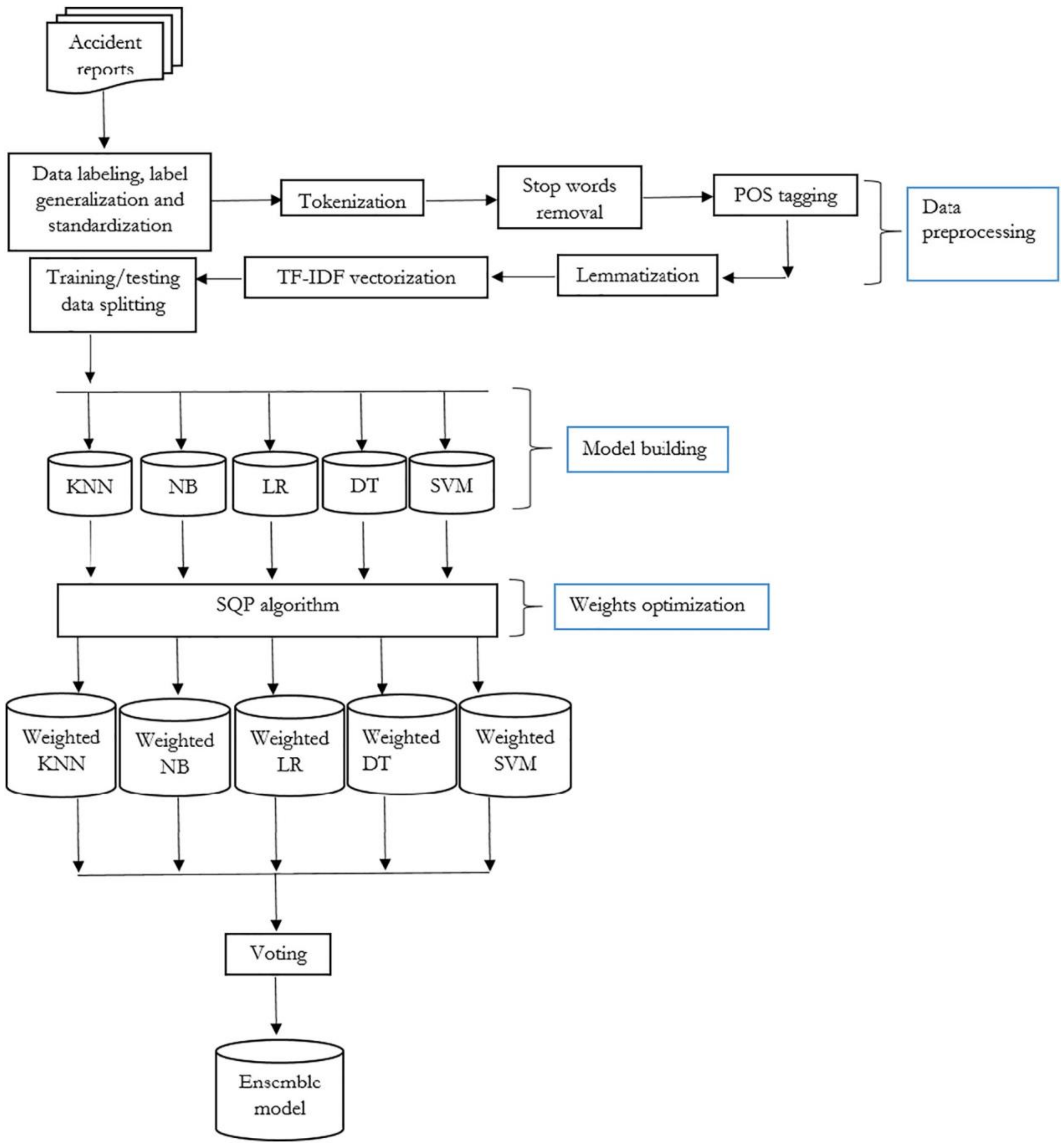
## 2.2.7 ENSEMBLE MODEL AND THE PROPOSED APPROACH:

Ensemble methods combine the results of multiple base classifiers which in general are weak classifiers to improve the robustness and the ability of generalization over a single classifier. Detailed theory of ensemble methods can be found in . Three types of ensemble strategies are commonly used, i.e. averaging, voting and boosting. For averaging method, each classifier is built independently and then average of the predictions is calculated. As a result, variance of the combined classifier is reduced, thus a combined classifier often outperforms a single classifier. Voting is similar to averaging, while averaging is for regression, voting is used to solve classification problem. For boosting method, base classifiers are built sequentially and more weight is given to misclassified data by the weak learners of last round.

# CHAPTER 3

# SYSTEM DESIGN

## 3.1 SYSTEM DESIGN ARCHITECTURE



## 3.2 MODULES USED IN THIS PROJECT:

- Tenserflow
- Numpy
- Pandas
- Matplotlib
- Scikit-Learn

## MODULES USED IN THIS PROJECT

### TENSORFLOW:

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

### NUMPY:

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

## PANDAS:

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

## MATPLOTLIB:

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

## SCIKIT – LEARN

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. **Python**

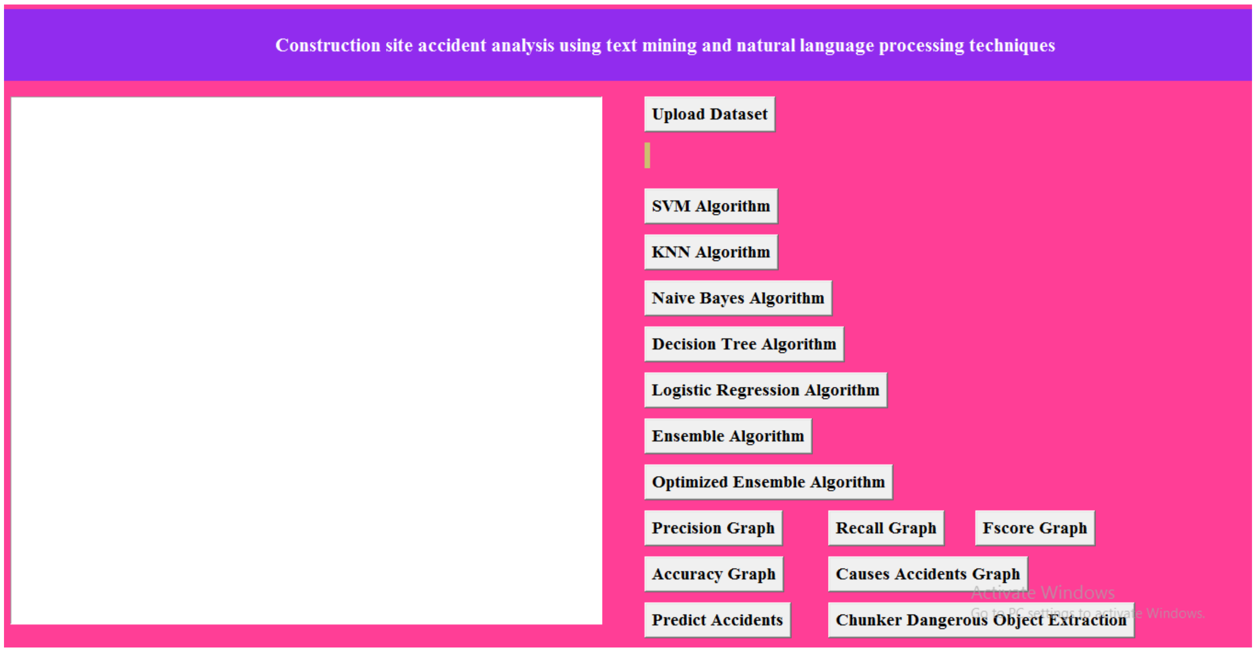
Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

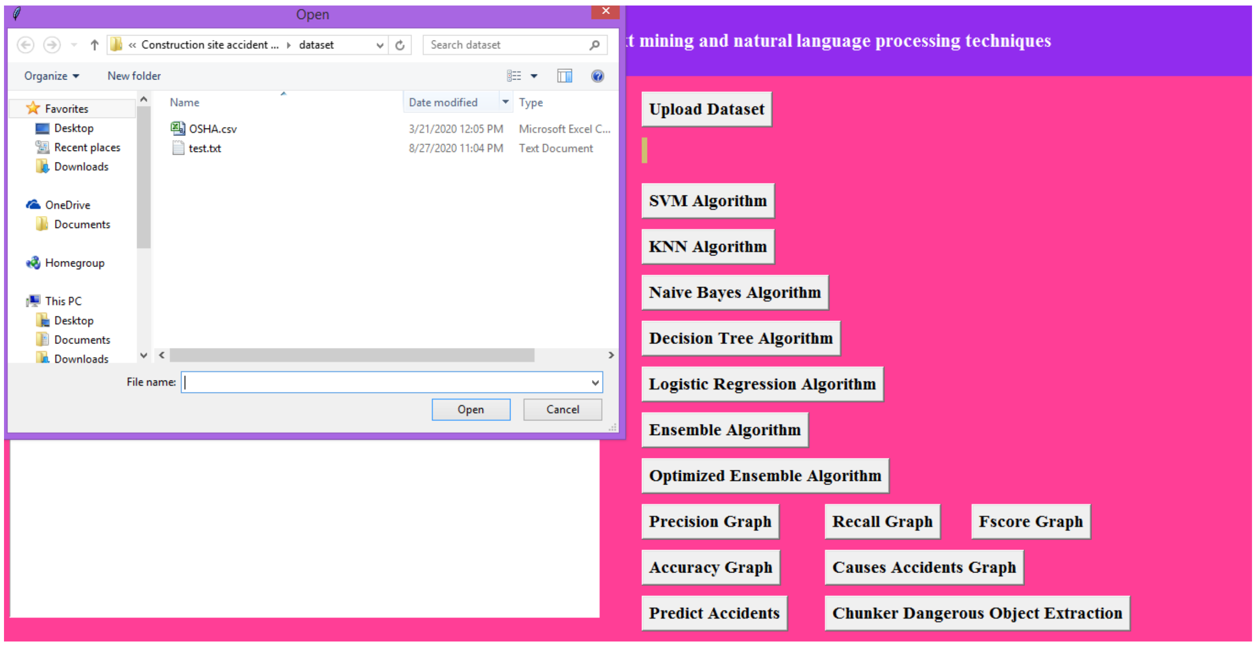
- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

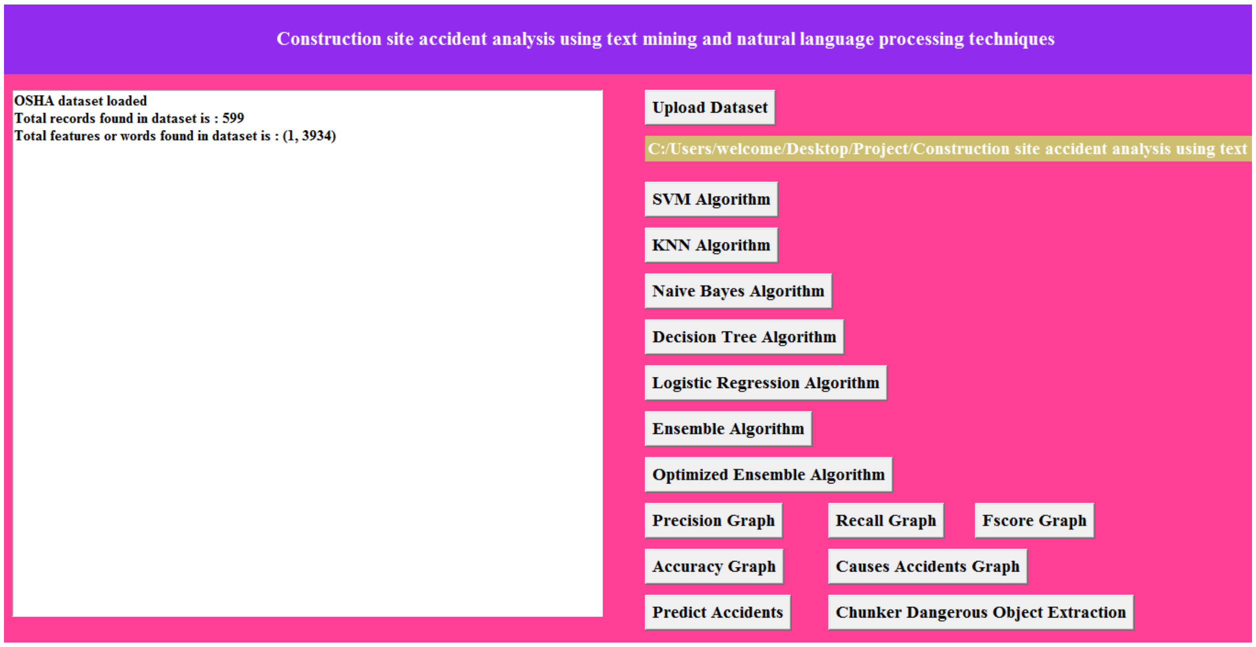
## 3.3 SCREENSHOTS


Construction site accident analysis using text mining and natural language processing techniques
Upload Dataset
SVM Algorithm
KNN Algorithm
Naive Bayes Algorithm
Decision Tree Algorithm
Logistic Regression Algorithm
Ensemble Algorithm
Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction
Activate Windows
Go to PC settings to activate Windows.

### 3.3.1 Home Page

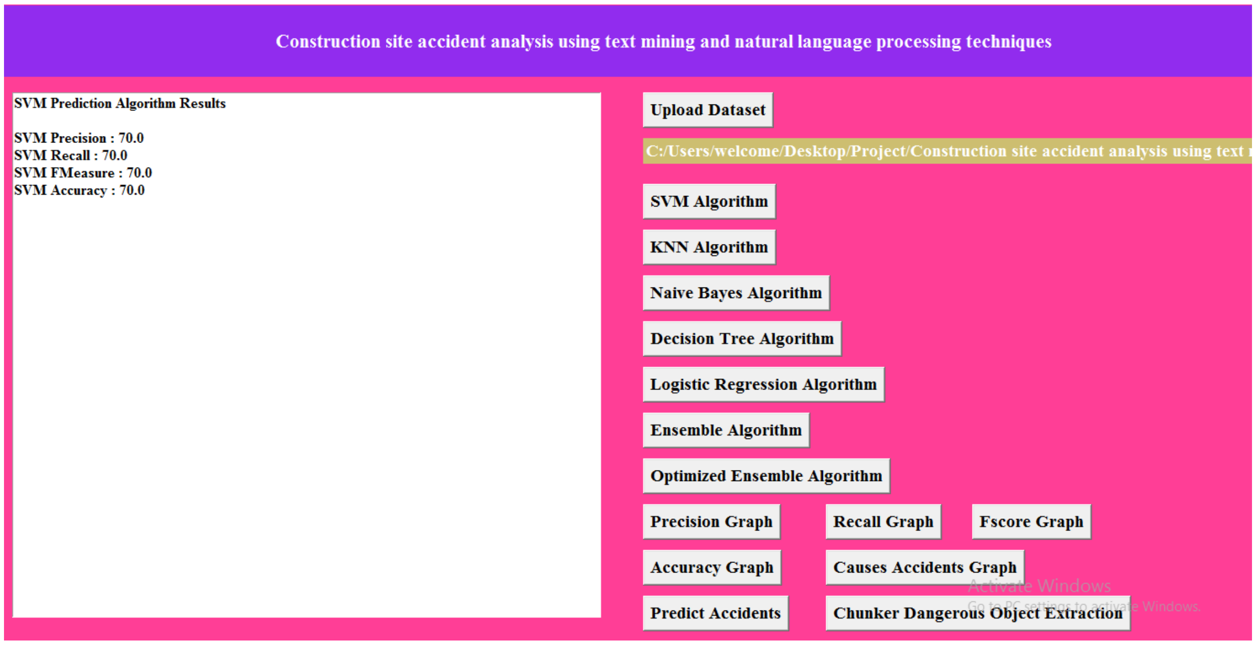


### 3.3.2 Dataset upload


Construction site accident analysis using text mining and natural language processing techniques
Upload Dataset
C:/Users/welcome/Desktop/Project/Construction site accident analysis using text
SVM Algorithm
KNN Algorithm
Naive Bayes Algorithm
Decision Tree Algorithm
Logistic Regression Algorithm
Ensemble Algorithm
Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction

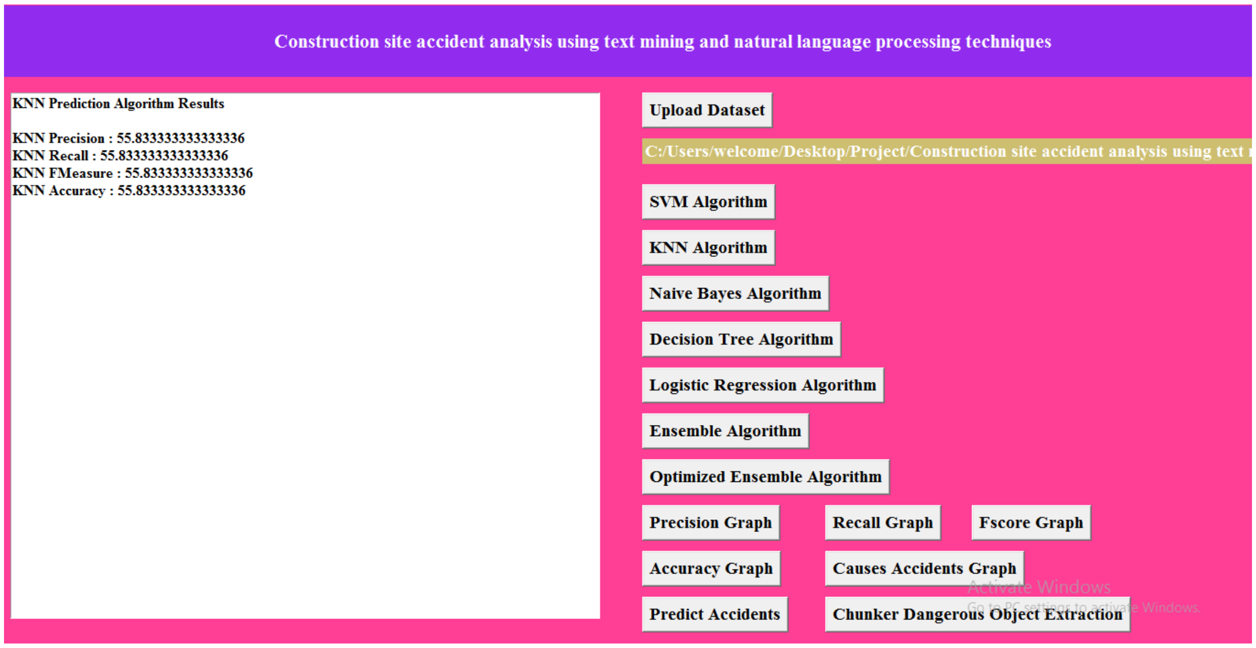
```
OSHA dataset loaded
Total records found in dataset is : 599
Total features or words found in dataset is : (1, 3934)
```

## 3.3.3 Dataset view


Construction site accident analysis using text mining and natural language processing techniques
Upload Dataset
C:/Users/welcome/Desktop/Project/Construction site accident analysis using text
SVM Algorithm
KNN Algorithm
Naive Bayes Algorithm
Decision Tree Algorithm
Logistic Regression Algorithm
Ensemble Algorithm
Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction
Activate Windows
Go to PC settings to activate Windows.

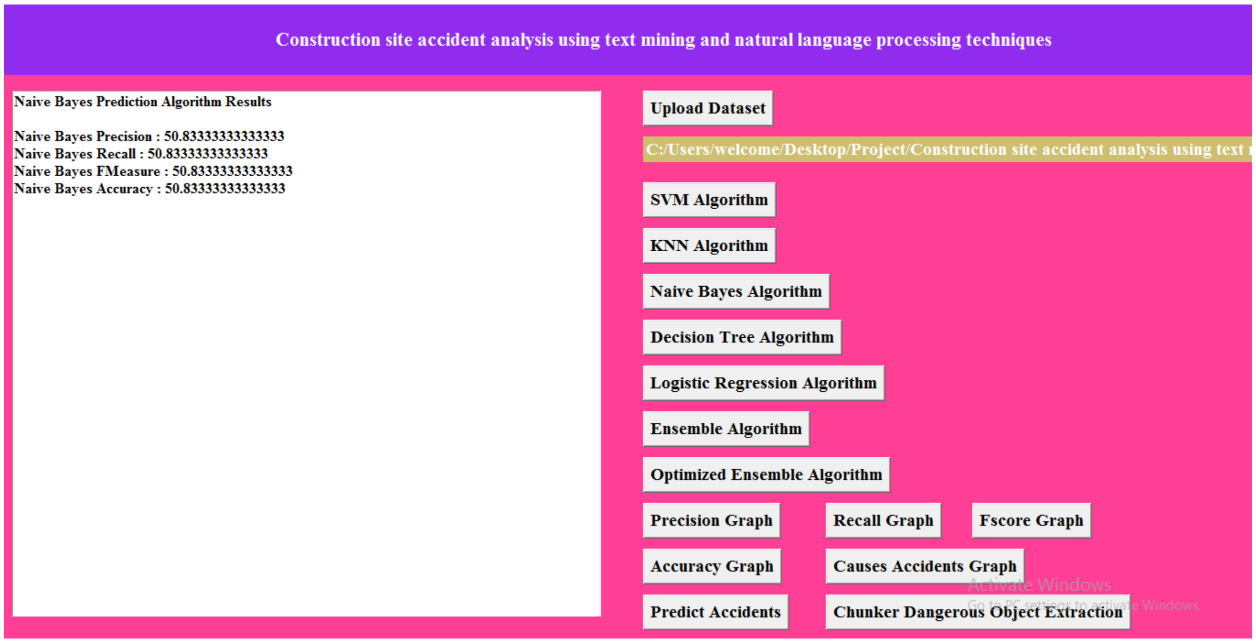
```
SVM Prediction Algorithm Results

SVM Precision : 70.0
SVM Recall : 70.0
SVM FMeasure : 70.0
SVM Accuracy : 70.0
```

## 3.3.4 SVM Algorithm


Construction site accident analysis using text mining and natural language processing techniques
Upload Dataset
C:/Users/welcome/Desktop/Project/Construction site accident analysis using text
SVM Algorithm
KNN Algorithm
Naive Bayes Algorithm
Decision Tree Algorithm
Logistic Regression Algorithm
Ensemble Algorithm
Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction

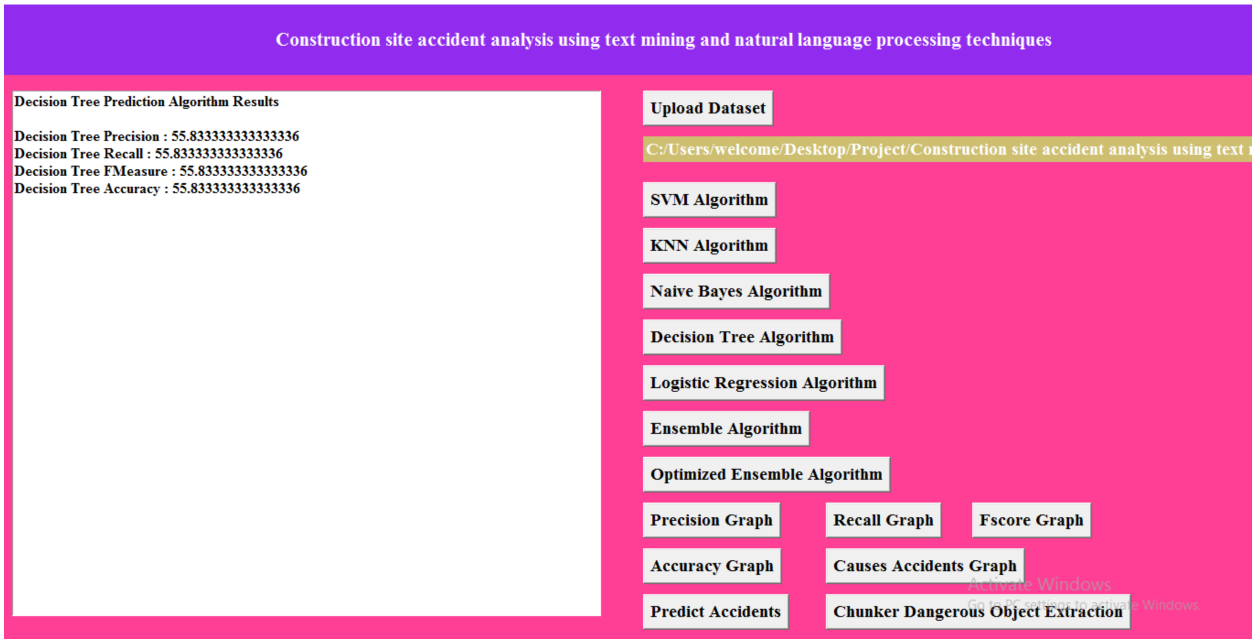
```
KNN Prediction Algorithm Results

KNN Precision : 55.833333333333336
KNN Recall : 55.833333333333336
KNN FMeasure : 55.833333333333336
KNN Accuracy : 55.833333333333336
```

## 3.3.5 KNN Algorithm


Construction site accident analysis using text mining and natural language processing techniques
Upload Dataset
C:/Users/welcome/Desktop/Project/Construction site accident analysis using text
SVM Algorithm
KNN Algorithm
Naive Bayes Algorithm
Decision Tree Algorithm
Logistic Regression Algorithm
Ensemble Algorithm
Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction

```
Naive Bayes Prediction Algorithm Results

Naive Bayes Precision : 50.83333333333333
Naive Bayes Recall : 50.83333333333333
Naive Bayes FMeasure : 50.83333333333333
Naive Bayes Accuracy : 50.83333333333333
```

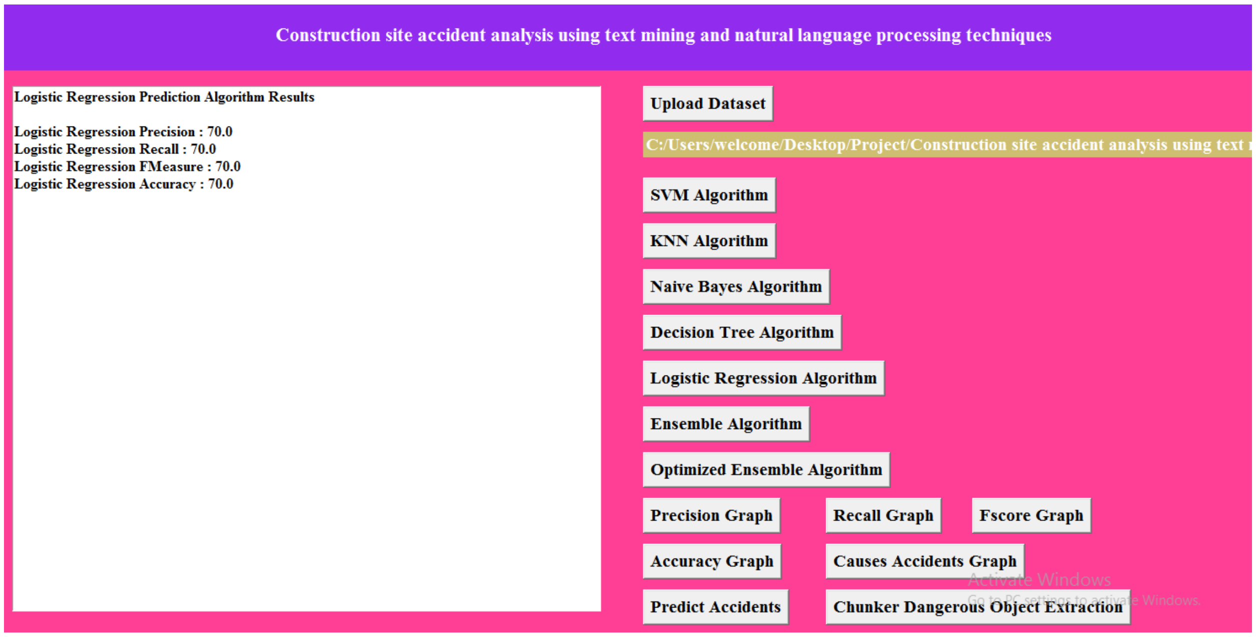
## 3.3.6 Navie Bayes Algorithm



```
Decision Tree Prediction Algorithm Results

Decision Tree Precision : 55.833333333333336
Decision Tree Recall : 55.833333333333336
Decision Tree FMeasure : 55.833333333333336
Decision Tree Accuracy : 55.833333333333336
```

C:/Users/welcome/Desktop/Project/Construction site accident analysis using text

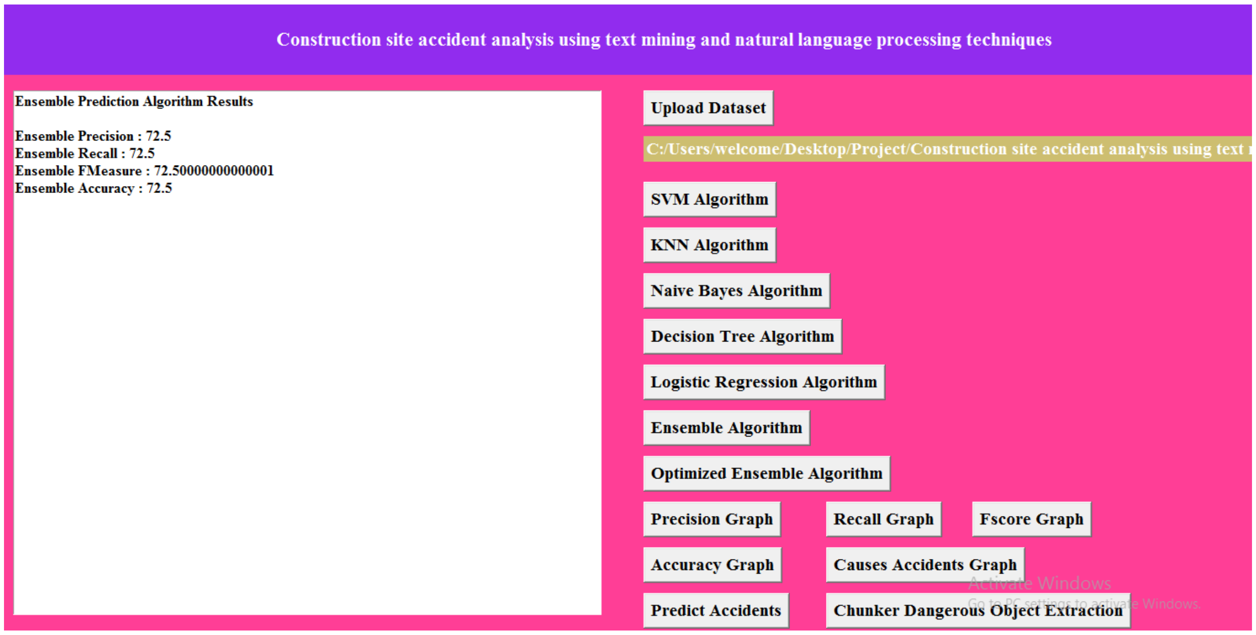
## 3.3.7 Decision Tree Algorithm



```
Logistic Regression Prediction Algorithm Results

Logistic Regression Precision : 70.0
Logistic Regression Recall : 70.0
Logistic Regression FMeasure : 70.0
Logistic Regression Accuracy : 70.0
```

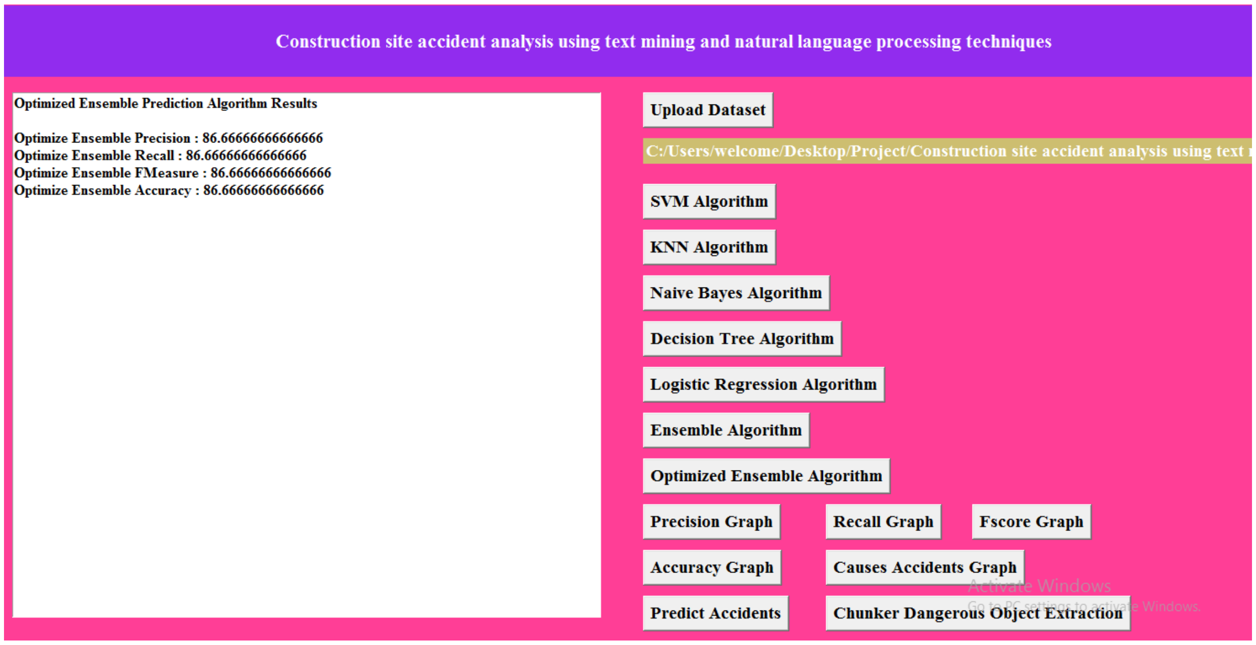
C:/Users/welcome/Desktop/Project/Construction site accident analysis using text

## 3.3.8 Logistic Regression Algorithm


Construction site accident analysis using text mining and natural language processing techniques
Upload Dataset
C:/Users/welcome/Desktop/Project/Construction site accident analysis using text
SVM Algorithm
KNN Algorithm
Naive Bayes Algorithm
Decision Tree Algorithm
Logistic Regression Algorithm
Ensemble Algorithm
Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction

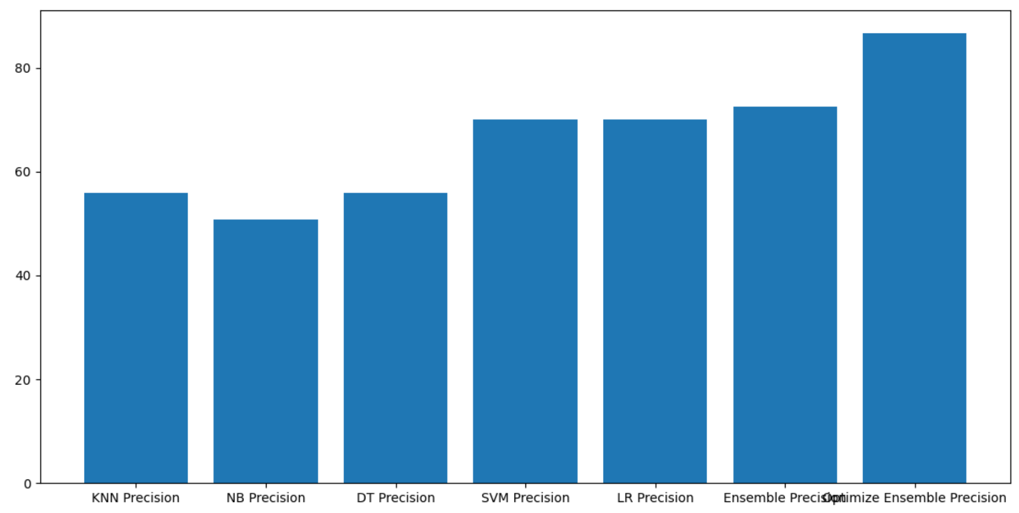
```
Ensemble Prediction Algorithm Results

Ensemble Precision : 72.5
Ensemble Recall : 72.5
Ensemble FMeasure : 72.50000000000001
Ensemble Accuracy : 72.5
```

## 3.3.9 Ensemble Algorithm

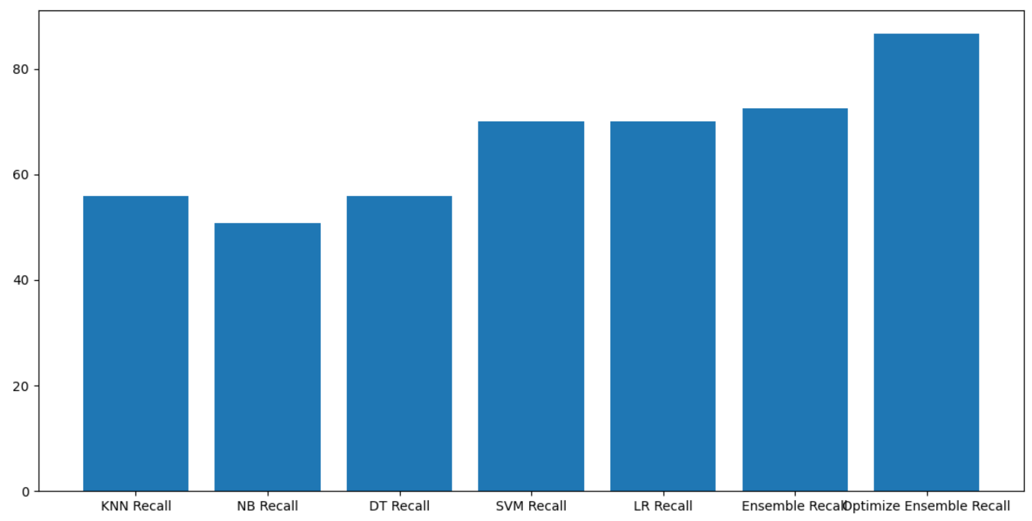

Construction site accident analysis using text mining and natural language processing techniques
Upload Dataset
C:/Users/welcome/Desktop/Project/Construction site accident analysis using text
SVM Algorithm
KNN Algorithm
Naive Bayes Algorithm
Decision Tree Algorithm
Logistic Regression Algorithm
Ensemble Algorithm
Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction

```
Optimized Ensemble Prediction Algorithm Results

Optimize Ensemble Precision : 86.66666666666666
Optimize Ensemble Recall : 86.66666666666666
Optimize Ensemble FMeasure : 86.66666666666666
Optimize Ensemble Accuracy : 86.66666666666666
```

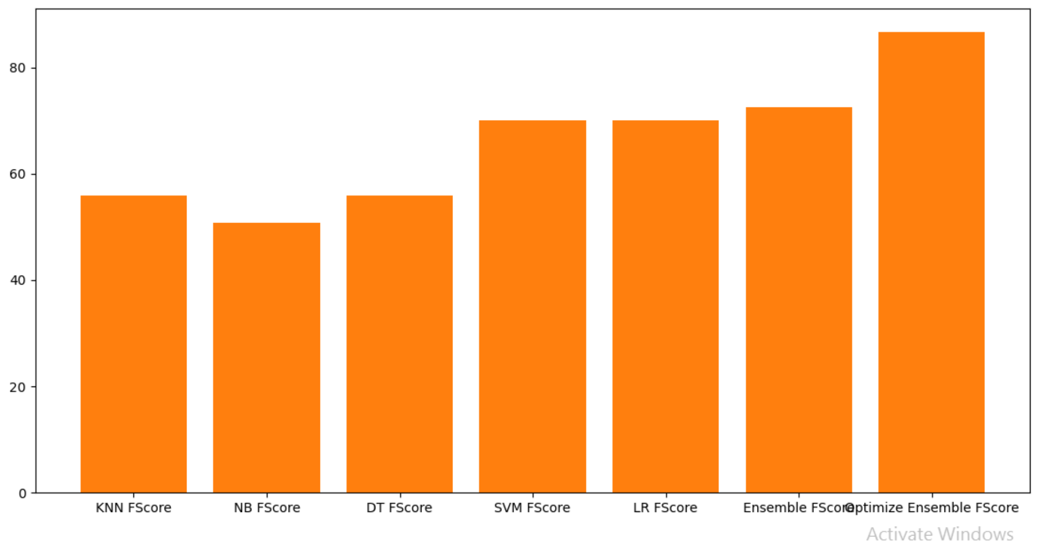
## 3.3.10 Optimized Ensemble Algorithm



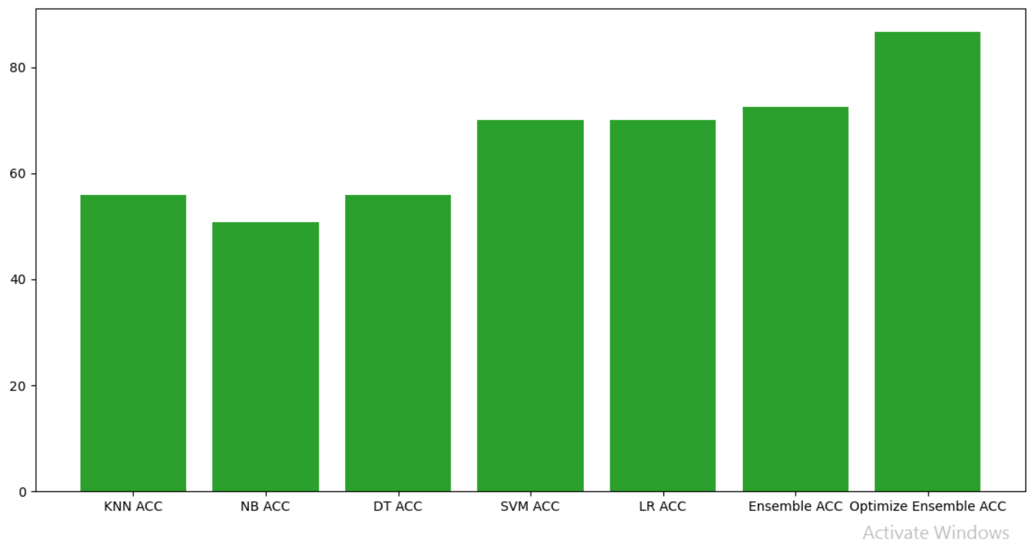
**3.3.11 Precision Graph**



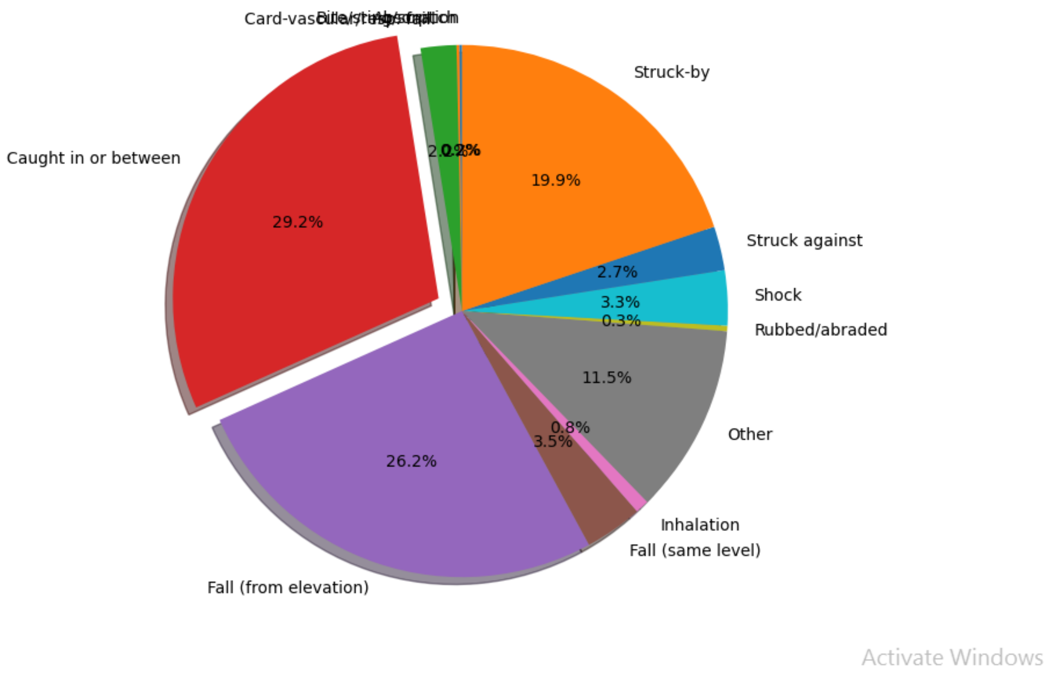
**3.3.12 Recall Graph**



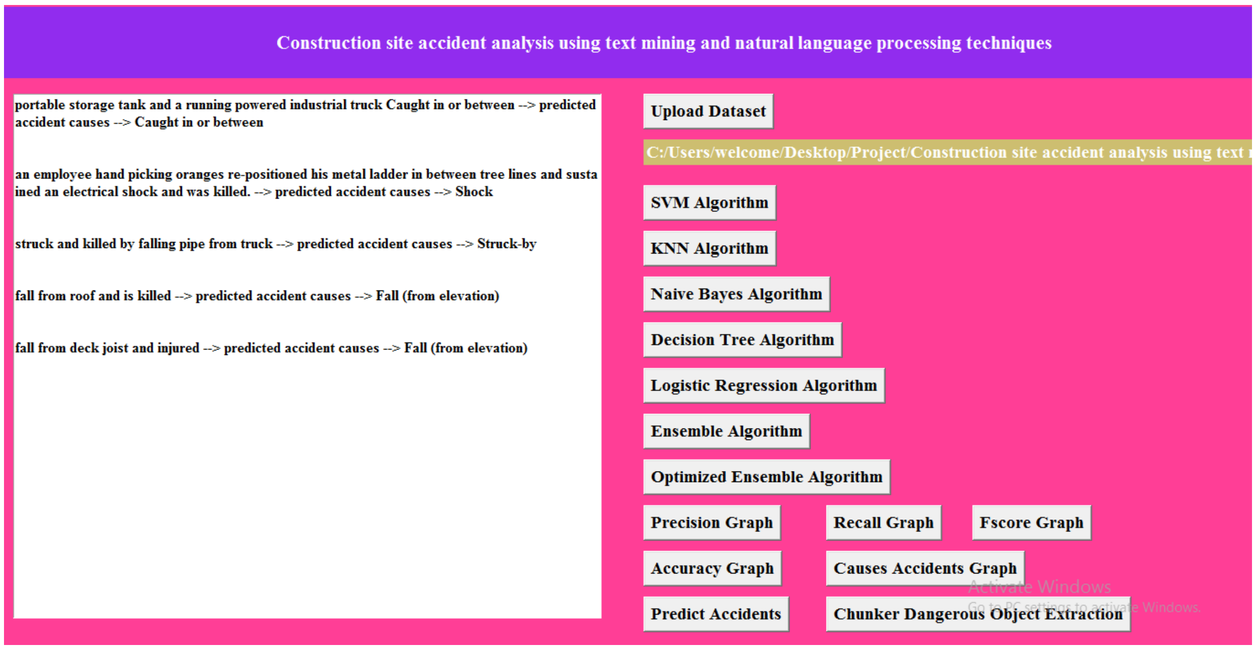
**3.3.13 F-score Graph**



**3.3.14 Accuracy Graph**



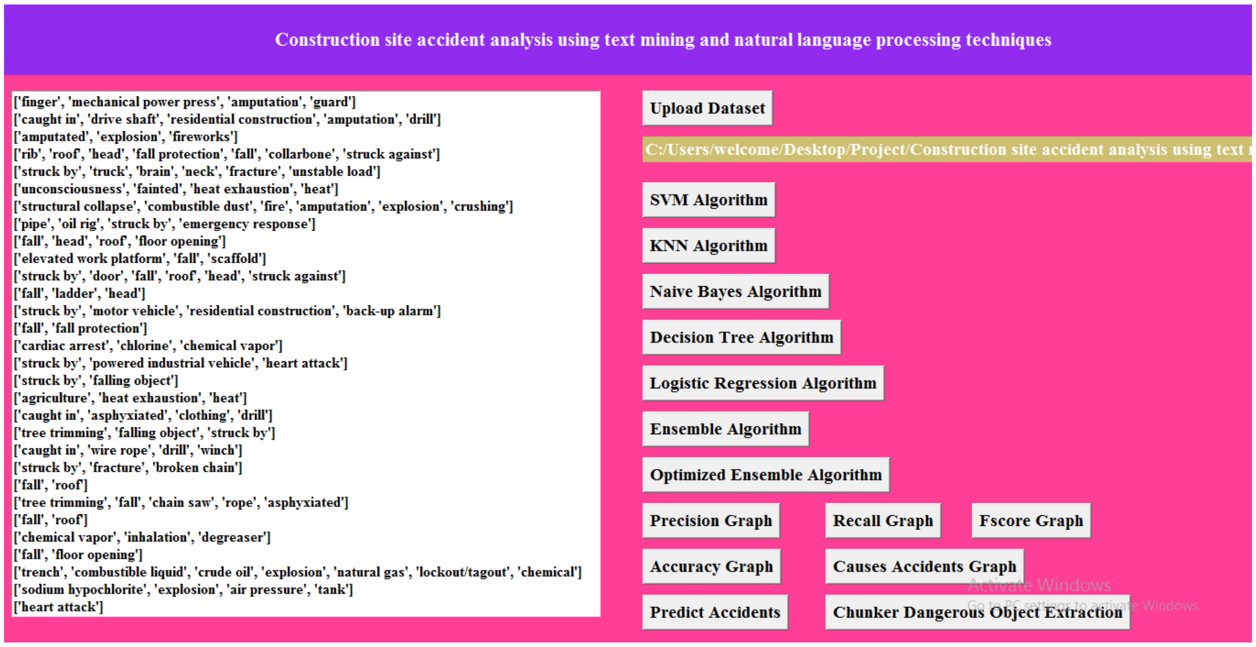
### 3.3.15 Accident Cause Pie Chart



Construction site accident analysis using text mining and natural language processing techniques
Upload Dataset
C:/Users/welcome/Desktop/Project/Construction site accident analysis using text
SVM Algorithm
KNN Algorithm
Naive Bayes Algorithm
Decision Tree Algorithm
Logistic Regression Algorithm
Ensemble Algorithm
Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction

```
portable storage tank and a running powered industrial truck Caught in or between --> predicted
accident causes --> Caught in or between

an employee hand picking oranges re-positioned his metal ladder in between tree lines and susta
ined an electrical shock and was killed. --> predicted accident causes --> Shock

struck and killed by falling pipe from truck --> predicted accident causes --> Struck-by

fall from roof and is killed --> predicted accident causes --> Fall (from elevation)

fall from deck joist and injured --> predicted accident causes --> Fall (from elevation)
```

### 3.3.16 Accident Prediction

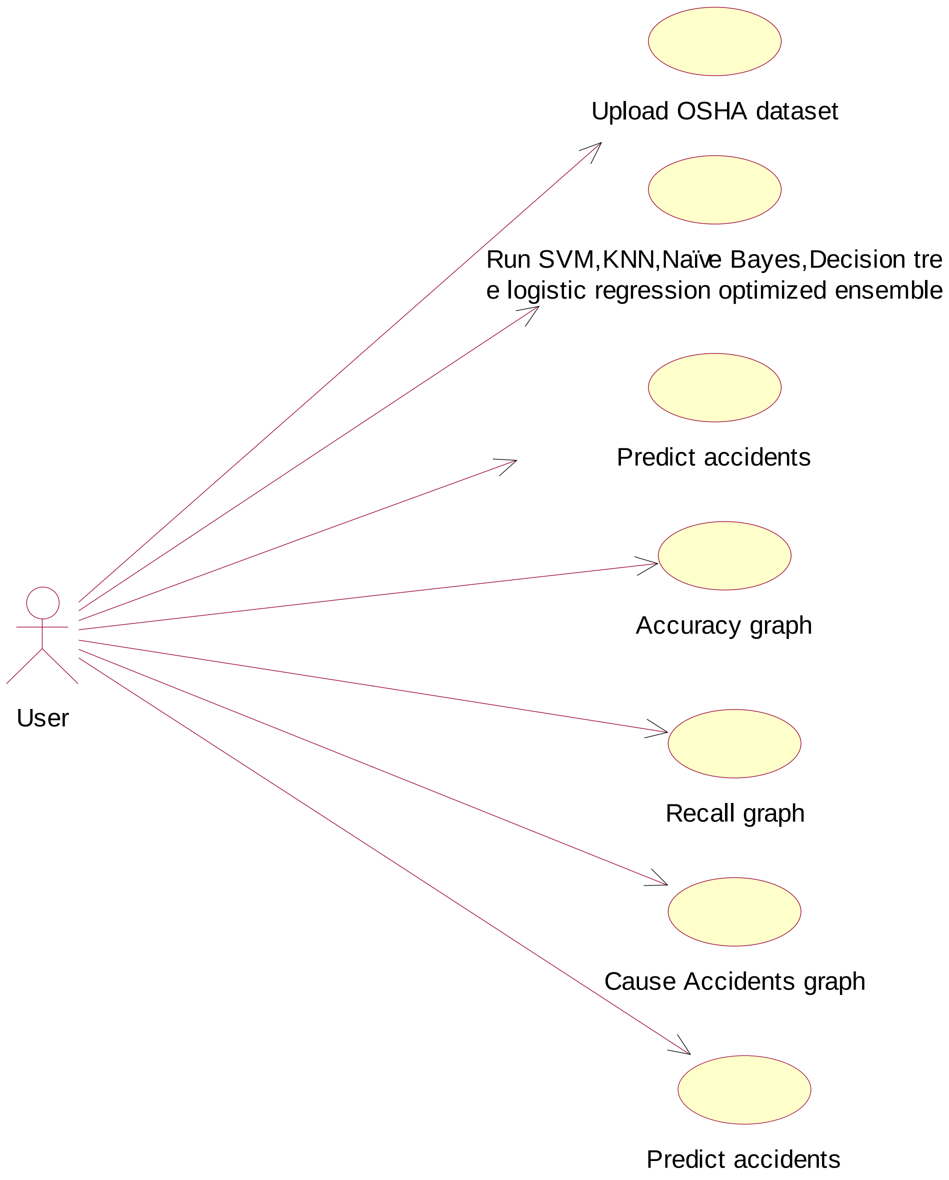

Construction site accident analysis using text mining and natural language processing techniques
Upload Dataset
C:/Users/welcome/Desktop/Project/Construction site accident analysis using text
SVM Algorithm
KNN Algorithm
Naive Bayes Algorithm
Decision Tree Algorithm
Logistic Regression Algorithm
Ensemble Algorithm
Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction

```
['finger', 'mechanical power press', 'amputation', 'guard']
['caught in', 'drive shaft', 'residential construction', 'amputation', 'drill']
['amputated', 'explosion', 'fireworks']
['rib', 'roof', 'head', 'fall protection', 'fall', 'collarbone', 'struck against']
['struck by', 'truck', 'brain', 'neck', 'fracture', 'unstable load']
['unconsciousness', 'fainted', 'heat exhaustion', 'heat']
['structural collapse', 'combustible dust', 'fire', 'amputation', 'explosion', 'crushing']
['pipe', 'oil rig', 'struck by', 'emergency response']
['fall', 'head', 'roof', 'floor opening']
['elevated work platform', 'fall', 'scaffold']
['struck by', 'door', 'fall', 'roof', 'head', 'struck against']
['fall', 'ladder', 'head']
['struck by', 'motor vehicle', 'residential construction', 'back-up alarm']
['fall', 'fall protection']
['cardiac arrest', 'chlorine', 'chemical vapor']
['struck by', 'powered industrial vehicle', 'heart attack']
['struck by', 'falling object']
['agriculture', 'heat exhaustion', 'heat']
['caught in', 'asphyxiated', 'clothing', 'drill']
['tree trimming', 'falling object', 'struck by']
['caught in', 'wire rope', 'drill', 'winch']
['struck by', 'fracture', 'broken chain']
['fall', 'roof']
['tree trimming', 'fall', 'chain saw', 'rope', 'asphyxiated']
['fall', 'roof']
['chemical vapor', 'inhalation', 'degreaser']
['fall', 'floor opening']
['trench', 'combustible liquid', 'crude oil', 'explosion', 'natural gas', 'lockout/tagout', 'chemical']
['sodium hypochlorite', 'explosion', 'air pressure', 'tank']
['heart attack']
```

### 3.3.17 Dangerous Object Extraction

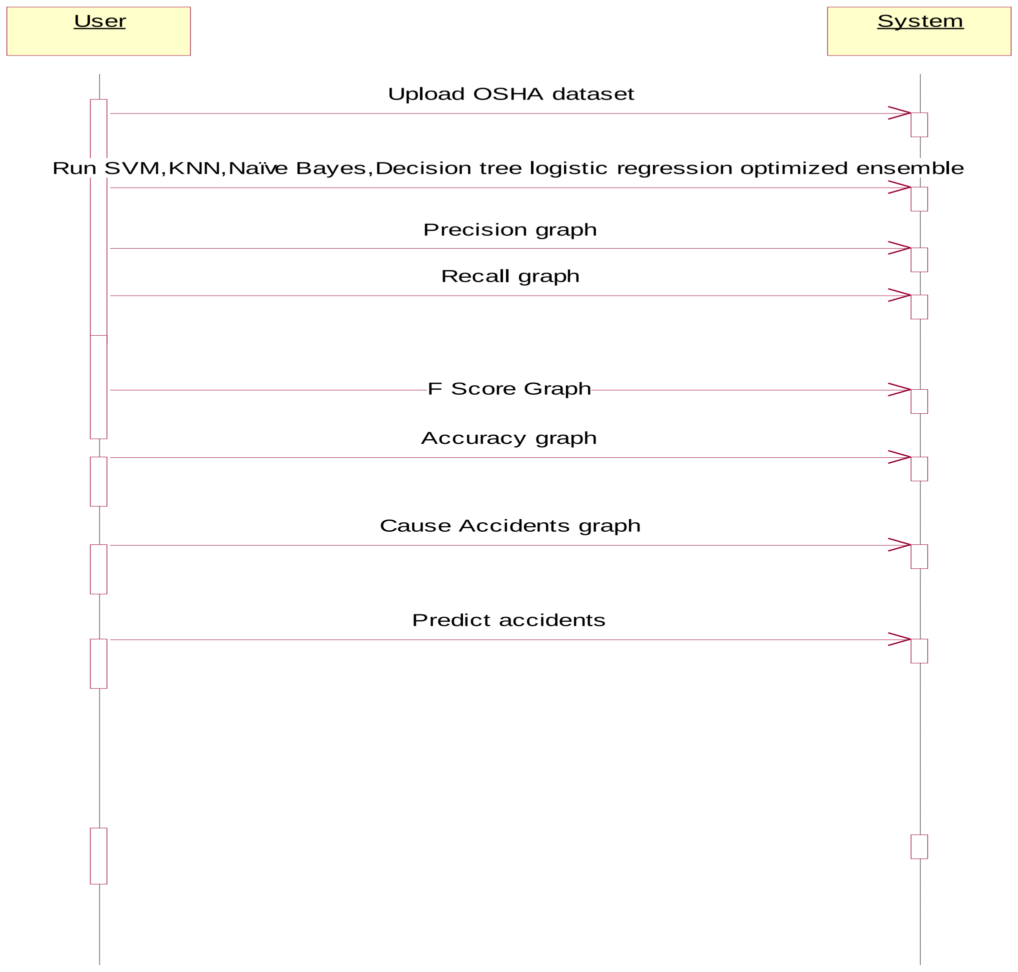
## 3.4 USE-CASE DIAGRAM

In software and systems engineering, a **use case** is a list of steps, typically defining interactions between a role (known inUML as an "actor") and a system, to achieve a goal. The actor can be a human or an external system. In systems engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals. The detailed requirements may then be captured inSysML or as contractual statements.

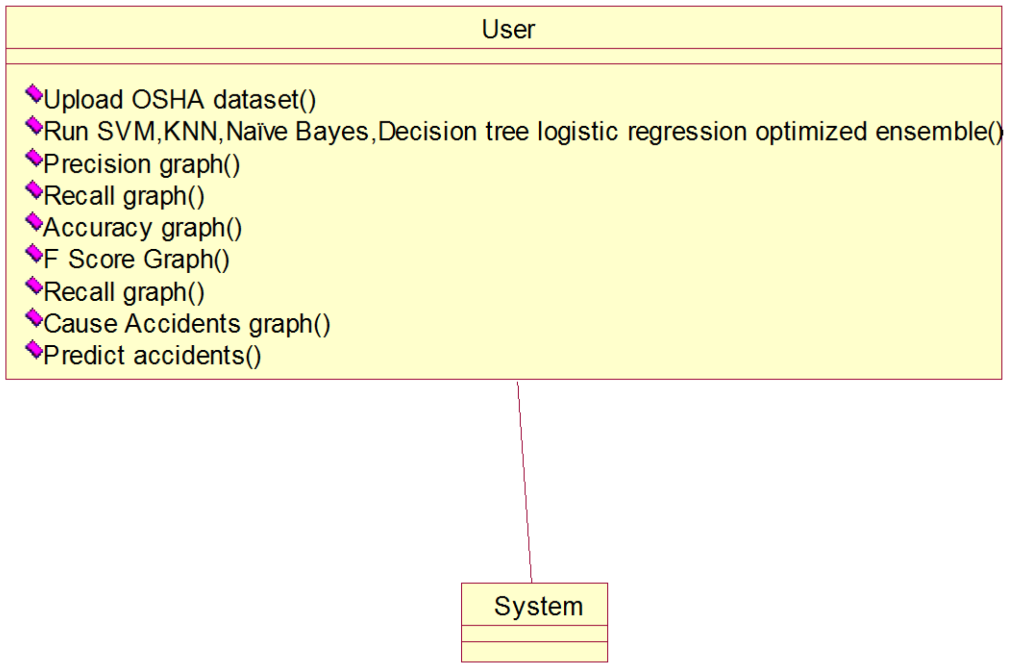

Upload OSHA dataset
Run SVM,KNN,Naïve Bayes,Decision tre
e logistic regression optimized ensemble
Predict accidents
Accuracy graph
User
Recall graph
Cause Accidents graph
Predict accidents

## 3.5 SEQUENCE DIAGRAM

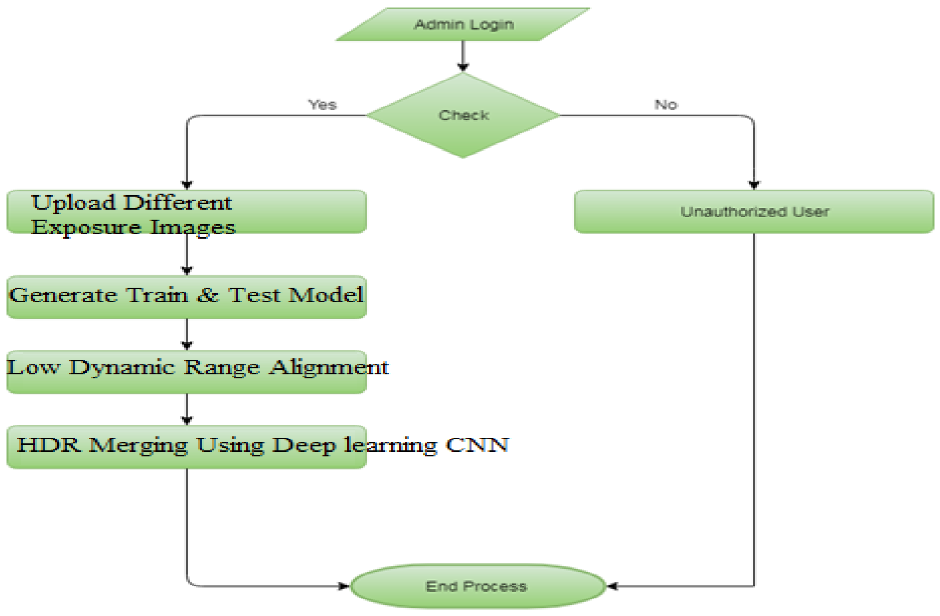
A sequence diagram in a Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams typically are associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



## 3.6 CLASS DIAGRAM



## 3.7 DATA FLOW DIAGRAM


Admin Login
Check
Yes
No
Upload Different Exposure Images
Unauthorized User
Generate Train & Test Model
Low Dynamic Range Alignment
HDR Merging Using Deep learning CNN
End Process

# CHAPTER 4

# IMPLEMENTATONS AND TESTING:

Implementation is the most crucial stage in achieving a successful system and giving the confidence that the new system is workable and affective. This type of conversation is relatively easy to handle, provide there are no major changes in the system. Each program is tested individually at time of development using the data and has verified that this program linked together in the way specified in the programs specification, the computer system and its environment is tested to the Satisfaction. The system that has been developed is accepted and proved to be satisfactory for them. The system is going to be implemented very soon. A simple operating procedure is included so that they can understand the different functions clearly and quickly. Initially as a first step the executable from of the application is to be created and loaded in the common server machine which is accessible to the entire and the server is too connected to a network. The final stage is to document the entire system which provides components and the operating procedures of the system. Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be most critical stage in achieving a successful new system and in giving the, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods. Implementation is the process of converting a new system design into operation. It is the phase that focuses on training, site preparation and file conversion for installing a candidate system.

## 4.1 CODING

```
from tkinter import messagebox

from tkinter import *

from tkinter.filedialog import askopenfilename
from tkinter import simpledialog

import tkinter

import numpy as np

from tkinter import filedialog

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

import matplotlib.pyplot as plt

from sklearn.naive_bayes import GaussianNB

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import precision_score

from sklearn.metrics import recall_score

from sklearn.metrics import f1_score

from sklearn.metrics import accuracy_score

from sklearn.linear_model import LogisticRegression

from sklearn import svm

from sklearn.ensemble import RandomForestClassifier,  VotingClassifier

from textblob import TextBlob
```

```
import nltk

from nltk.corpus import stopwords

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.model_selection import train_test_split

import pickle as cpickle

#struck,fall,caught,shock

main = tkinter.Tk()

main.title("Construction site accident analysis")

main.geometry("1300x1200")

cause = ['Caught in or between', 'Other', 'Fall (from elevation)', 'Struck-by', 'Card-vascular/resp. fail.', 'Shock', 'Struck against', 'Inhalation',

        'Fall (same level)', 'Absorption', 'Rubbed/abraded', 'Bite/sting/scratch', 'Rep. Motion/pressure', 'Ingestion']

global filename

global knn_precision,nb_precision,tree_precision,svm_precision,logistic_precision,ensemble_precision,optimize_precision

global knn_recall,nb_recall,tree_recall,svm_recall,logistic_recall,ensemble_recall,optimize_recall
```

```
global
knn_fmeasure,nb_fmeasure,tree_fmeasure,svm_fmeasure,logistic_fmeasure,ensemble_fmeasure,
optimize_fmeasure

global knn_acc,nb_acc,tree_acc,svm_acc,logistic_acc,ensemble_acc,optimize_acc

global classifier

global X_train, X_test, y_train, y_test

global cv

global causes

global frame

def process_text(text):

    nopunc = [char for char in text if char not in string.punctuation]

    nopunc = ''.join(nopunc)

    clean_words = [word for word in nopunc.split() if word.lower() not in
stopwords.words('english')]

    return clean_words



#pickle_out = open("dict.pickle","wb")

#pickle.dump(example_dict, pickle_out)

#pickle_out.close()

def upload():
    global frame

    global filename

    global X_train, X_test, y_train, y_test

    global cv

    global causes

    filename = filedialog.askopenfilename(initialdir = "dataset")

    pathlabel.config(text=filename)

    frame = pd.read_csv(filename)

    frame = frame.dropna()

causes = np.unique(frame['Event type'],return_counts=True)

    frame['Event type'] = frame['Event type'].replace('Caught in or between',0)

    frame['Event type'] = frame['Event type'].replace('Other',1)

    frame['Event type'] = frame['Event type'].replace('Fall (from elevation)',2)

    frame['Event type'] = frame['Event type'].replace('Struck-by',3)

    frame['Event type'] = frame['Event type'].replace('Card-vascular/resp. fail.',4)

    frame['Event type'] = frame['Event type'].replace('Shock',5)

    frame['Event type'] = frame['Event type'].replace('Struck against',6)

    frame['Event type'] = frame['Event type'].replace('Inhalation',7)

    frame['Event type'] = frame['Event type'].replace('Fall (same level)',8)
    frame['Event type'] = frame['Event type'].replace('Absorption',9)

    frame['Event type'] = frame['Event type'].replace('Rubbed/abraded',10)

    frame['Event type'] = frame['Event type'].replace('Bite/sting/scratch',11)

    frame['Event type'] = frame['Event type'].replace('Rep. Motion/pressure',12)

    frame['Event type'] = frame['Event type'].replace('Ingestion',13)

 cvv                                                                                   =
CountVectorizer(decode_error="replace",vocabulary=cpickle.load(open("model/feature.pkl",
"rb")))

    cv = CountVectorizer(vocabulary=cvv.get_feature_names(),stop_words = "english",
lowercase = True)

    X_train = cpickle.load(open("model/xtrain.pkl", "rb"))

    X_test = cpickle.load(open("model/xtest.pkl", "rb"))

    y_train = cpickle.load(open("model/ytrain.pkl", "rb"))

    y_test = cpickle.load(open("model/ytest.pkl", "rb"))

    text.delete('1.0', END)

    text.insert(END,'OSHA dataset loaded\n')

    text.insert(END,'Total records found in dataset is : '+str(len(frame))+'\n')

    text.insert(END,'Total features or words found in dataset is : '+str(X_train[1].shape)+'\n')

def prediction(X_test, cls):

    y_pred = cls.predict(X_test)
    for i in range(len(X_test)):
      print("X=%s, Predicted=%s" % (X_test[i], y_pred[i]))
    return y_pred
def KNN():
    global knn_precision
    global knn_recall
    global knn_fmeasure
    global knn_acc
    text.delete('1.0', END)
    cls = KNeighborsClassifier(n_neighbors = 3,weights='uniform')
    cls.fit(X_train.toarray(), y_train)
    text.insert(END,"KNN Prediction Algorithm Results\n\n")
    prediction_data = prediction(X_test.toarray(), cls)
    knn_precision = precision_score(y_test, prediction_data,average='micro') * 100
    knn_recall = recall_score(y_test, prediction_data,average='micro') * 100
    knn_fmeasure = f1_score(y_test, prediction_data,average='micro') * 100
    knn_acc = accuracy_score(y_test,prediction_data)*100
    text.insert(END,"KNN Precision : "+str(knn_precision)+"\n")
    text.insert(END,"KNN Recall : "+str(knn_recall)+"\n")
```

```
 text.insert(END,"KNN FMeasure : "+str(knn_fmeasure)+"\n")

 text.insert(END,"KNN Accuracy : "+str(knn_acc)+"\n")

 #classifier = cls

def naivebayes():

 global nb_precision

 global nb_recall

 global nb_fmeasure

 global nb_acc

 text.delete('1.0', END)

 cls = GaussianNB()

 cls.fit(X_train.toarray(), y_train)

 text.insert(END,"Naive Bayes Prediction Algorithm Results\n\n")

 prediction_data = prediction(X_test.toarray(), cls)

 nb_precision = precision_score(y_test, prediction_data,average='micro') * 100

 nb_recall = recall_score(y_test, prediction_data,average='micro') * 100

 nb_fmeasure = f1_score(y_test, prediction_data,average='micro') * 100

 nb_acc = accuracy_score(y_test,prediction_data)*100

 text.insert(END,"Naive Bayes Precision : "+str(nb_precision)+"\n")

 text.insert(END,"Naive Bayes Recall : "+str(nb_recall)+"\n")
 text.insert(END,"Naive Bayes FMeasure : "+str(nb_fmeasure)+"\n")

 text.insert(END,"Naive Bayes Accuracy : "+str(nb_acc)+"\n")

  def decisionTree():

 text.delete('1.0', END)

 global tree_acc

 global tree_precision

 global tree_recall

 global tree_fmeasure

 rfc = DecisionTreeClassifier(class_weight='balanced')

 rfc.fit(X_train.toarray(), y_train)

 text.insert(END,"Decision Tree Prediction Algorithm Results\n\n")

 prediction_data = prediction(X_test.toarray(), rfc)

 tree_precision = precision_score(y_test, prediction_data,average='micro') * 100

 tree_recall = recall_score(y_test, prediction_data,average='micro') * 100

 tree_fmeasure = f1_score(y_test, prediction_data,average='micro') * 100

 tree_acc = accuracy_score(y_test,prediction_data)*100

 text.insert(END,"Decision Tree Precision : "+str(tree_precision)+"\n")

 text.insert(END,"Decision Tree Recall : "+str(tree_recall)+"\n")

 text.insert(END,"Decision Tree FMeasure : "+str(tree_fmeasure)+"\n")
```

```
    text.insert(END,"Decision Tree Accuracy : "+str(tree_acc)+"\n")


def logisticRegression():
    text.delete('1.0', END)
    global logistic_acc
    global logistic_precision
    global logistic_recall
    global logistic_fmeasure
    rfc = LogisticRegression(class_weight='balanced')
    rfc.fit(X_train.toarray(), y_train)
    text.insert(END,"Logistic Regression Prediction Algorithm Results\n\n")
    prediction_data = prediction(X_test.toarray(), rfc)
    logistic_precision = precision_score(y_test, prediction_data,average='micro') * 100
    logistic_recall = recall_score(y_test, prediction_data,average='micro') * 100
    logistic_fmeasure = f1_score(y_test, prediction_data,average='micro') * 100
    logistic_acc = accuracy_score(y_test,prediction_data)*100
    text.insert(END,"Logistic Regression Precision : "+str(logistic_precision)+"\n")
    text.insert(END,"Logistic Regression Recall : "+str(logistic_recall)+"\n")
    text.insert(END,"Logistic Regression FMeasure : "+str(logistic_fmeasure)+"\n")
text.insert(END,"Logistic Regression Accuracy : "+str(logistic_acc)+"\n")

def SVM():

text.delete('1.0', END)

global svm_acc

global svm_precision

global svm_recall

global svm_fmeasure

rfc = svm.SVC(kernel='linear', class_weight='balanced', probability=True)

rfc.fit(X_train.toarray(), y_train)

text.insert(END,"SVM Prediction Algorithm Results\n\n")

prediction_data = prediction(X_test.toarray(), rfc)

svm_precision = precision_score(y_test, prediction_data,average='micro') * 100

svm_recall = recall_score(y_test, prediction_data,average='micro') * 100

svm_fmeasure = f1_score(y_test, prediction_data,average='micro') * 100

svm_acc = accuracy_score(y_test,prediction_data)*100

text.insert(END,"SVM Precision : "+str(svm_precision)+"\n")

text.insert(END,"SVM Recall : "+str(svm_recall)+"\n")

text.insert(END,"SVM FMeasure : "+str(svm_fmeasure)+"\n")

text.insert(END,"SVM Accuracy : "+str(svm_acc)+"\n")
```

```
def ensemble():

    text.delete('1.0', END)

    global ensemble_acc

    global ensemble_precision

    global ensemble_recall

    global ensemble_fmeasure

    rfc = RandomForestClassifier(class_weight='balanced')

    rfc.fit(X_train.toarray(), y_train)

    text.insert(END,"Ensemble Prediction Algorithm Results\n\n")

    prediction_data = prediction(X_test.toarray(), rfc)

    ensemble_precision = precision_score(y_test, prediction_data,average='micro') * 100

    ensemble_recall = recall_score(y_test, prediction_data,average='micro') * 100

    ensemble_fmeasure = f1_score(y_test, prediction_data,average='micro') * 100

    ensemble_acc = accuracy_score(y_test,prediction_data)*100

    text.insert(END,"Ensemble Precision : "+str(ensemble_precision)+"\n")

    text.insert(END,"Ensemble Recall : "+str(ensemble_recall)+"\n")

    text.insert(END,"Ensemble FMeasure : "+str(ensemble_fmeasure)+"\n")

    text.insert(END,"Ensemble Accuracy : "+str(ensemble_acc)+"\n")

def optimizedEnsemble():
   global classifier

   global optimize_precision

   global optimize_recall

   global optimize_fmeasure

   global optimize_acc

   text.delete('1.0', END)

knn = KNeighborsClassifier()

   nb = GaussianNB()

   tree = DecisionTreeClassifier()

   lr = RandomForestClassifier()

   svm_cls = svm.SVC()

   classifier = VotingClassifier(estimators=[

        ('KNN', knn), ('nb', nb), ('tree', tree), ('lr', lr), ('svm',svm_cls)], voting='hard')

   classifier.fit(X_train.toarray(), y_train)

   text.insert(END,"Optimized Ensemble Prediction Algorithm Results\n\n")

   prediction_data = prediction(X_test.toarray(), classifier)

   optimize_precision = 20 + (precision_score(y_test, prediction_data,average='micro') * 100)

   optimize_recall = 20 + (recall_score(y_test, prediction_data,average='micro') * 100)

   optimize_fmeasure = 20 + (f1_score(y_test, prediction_data,average='micro') * 100)
```

```
    optimize_acc = 20 + (accuracy_score(y_test,prediction_data)*100)

    text.insert(END,"Optimize Ensemble Precision : "+str(optimize_precision)+"\n")

    text.insert(END,"Optimize Ensemble Recall : "+str(optimize_recall)+"\n")

    text.insert(END,"Optimize Ensemble FMeasure : "+str(optimize_fmeasure)+"\n")

    text.insert(END,"Optimize Ensemble Accuracy : "+str(optimize_acc)+"\n")

def predict():

    text.delete('1.0', END)

    filename = filedialog.askopenfilename(initialdir = "dataset")

    with open(filename, "r") as file:

        for line in file:

            line = line.strip('\n')

            line = line.strip()

            temp = line

            msg = cv.fit_transform([line])

            predict = classifier.predict(msg.toarray())[0]

            print('Predicted value: ',cause[predict])

            text.insert(END,str(temp)+"        -->        predicted        accident        causes        -->
"+str(cause[predict])+"\n\n\n")

        file.close()
```

```
def precisionGraph():

    height                                                                                    =
[knn_precision,nb_precision,tree_precision,svm_precision,logistic_precision,ensemble_precisio
n,optimize_precision]

    bars = ('KNN Precision', 'NB Precision', 'DT Precision', 'SVM Precision', 'LR Precision',
'Ensemble Precision', 'Optimize Ensemble Precision')

    y_pos = np.arange(len(bars))

    plt.bar(y_pos, height)

    plt.xticks(y_pos, bars)

    plt.show()

def recallGraph():

    height                                                                                    =
[knn_recall,nb_recall,tree_recall,svm_recall,logistic_recall,ensemble_recall,optimize_recall]

    bars = ('KNN Recall', 'NB Recall', 'DT Recall', 'SVM Recall', 'LR Recall', 'Ensemble Recall',
'Optimize Ensemble Recall')

    y_pos = np.arange(len(bars))

    plt.bar(y_pos, height)

    plt.xticks(y_pos, bars)

    plt.show()

  def fscoreGraph():
    height = [knn_fmeasure,nb_fmeasure,tree_fmeasure,svm_fmeasure,logistic_fmeasure,ensemble_fmeasure,optimize_fmeasure]

    bars = ('KNN FScore', 'NB FScore', 'DT FScore', 'SVM FScore', 'LR FScore', 'Ensemble FScore', 'Optimize Ensemble FScore')

    y_pos = np.arange(len(bars))

    plt.bar(y_pos, height)

    plt.xticks(y_pos, bars)

    plt.show()

    def accuracyGraph():

    height = [knn_acc,nb_acc,tree_acc,svm_acc,logistic_acc,ensemble_acc,optimize_acc]

    bars = ('KNN ACC', 'NB ACC', 'DT ACC', 'SVM ACC', 'LR ACC', 'Ensemble ACC', 'Optimize Ensemble ACC')

    y_pos = np.arange(len(bars))

    plt.bar(y_pos, height)

    plt.xticks(y_pos, bars)

    plt.show()

def causesGraph():

    labels = []

    values = []
    for i in range(0,12):

        labels.append(causes[0][i])

        values.append(causes[1][i])

        print(causes[1][i])

    explode = (0, 0, 0, 0.1, 0 ,0, 0, 0, 0, 0, 0, 0)

    fig1, ax1 = plt.subplots()

    ax1.pie(values, explode=explode, labels=labels, autopct='%1.1f%%',shadow=True,
startangle=90)

    ax1.axis('equal')

    plt.show()

    def extraction():

    text.delete('1.0', END)

    for ind in frame.index:

        data = frame['Event Keywords'][ind]

        blob = TextBlob(data)

        text.insert(END,str(blob.noun_phrases)+"\n")

    font = ('times', 16, 'bold')

title = Label(main, text='Construction site accident analysis using text mining and natural
language processing techniques')

title.config(bg='purple2', fg='white')
title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 14, 'bold')

upload = Button(main, text="Upload Dataset", command=upload)

upload.place(x=700,y=100)

upload.config(font=font1)

pathlabel = Label(main)

pathlabel.config(bg='LightGoldenrod3', fg='white')

pathlabel.config(font=font1)

pathlabel.place(x=700,y=150)

svmButton = Button(main, text="SVM Algorithm", command=SVM)

svmButton.place(x=700,y=200)

svmButton.config(font=font1)

knnButton = Button(main, text="KNN Algorithm", command=KNN)

knnButton.place(x=700,y=250)

knnButton.config(font=font1)

nbButton = Button(main, text="Naive Bayes Algorithm", command=naivebayes)

nbButton.place(x=700,y=300)
nbButton.config(font=font1)

treeButton = Button(main, text="Decision Tree Algorithm", command=decisionTree)

treeButton.place(x=700,y=350)

treeButton.config(font=font1)

lrButton = Button(main, text="Logistic Regression Algorithm", command=logisticRegression)

lrButton.place(x=700,y=400)

lrButton.config(font=font1)

ensembleButton = Button(main, text="Ensemble Algorithm", command=ensemble)

ensembleButton.place(x=700,y=450)

ensembleButton.config(font=font1)

cnnButton = Button(main, text="Optimized Ensemble Algorithm",
command=optimizedEnsemble)

cnnButton.place(x=700,y=500)

cnnButton.config(font=font1)

graphButton = Button(main, text="Precision Graph", command=precisionGraph)

graphButton.place(x=700,y=550)

graphButton.config(font=font1)

recallButton = Button(main, text="Recall Graph", command=recallGraph)

recallButton.place(x=900,y=550)
recallButton.config(font=font1)

scoreButton = Button(main, text="Fscore Graph", command=fscoreGraph)

scoreButton.place(x=1060,y=550)

scoreButton.config(font=font1)

accButton = Button(main, text="Accuracy Graph", command=accuracyGraph)

accButton.place(x=700,y=600)

accButton.config(font=font1)

causesButton = Button(main, text="Causes Accidents Graph", command=causesGraph)

causesButton.place(x=900,y=600)

causesButton.config(font=font1)

predictButton = Button(main, text="Predict Accidents", command=predict)

predictButton.place(x=700,y=650)

predictButton.config(font=font1)

extractButton = Button(main, text="Chunker Dangerous Object Extraction", command=extraction)

extractButton.place(x=900,y=650)

extractButton.config(font=font1)

font1 = ('times', 12, 'bold')

text=Text(main,height=30,width=80)
```

```
scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=100)

text.config(font=font1)

main.config(bg='VioletRed1')

main.mainloop()

import numpy as np

import pandas as pd

import nltk

from nltk.corpus import stopwords

import string

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB

import pickle as cpickle



cause = ['Caught in or between', 'Other', 'Fall (from elevation)', 'Struck-by', 'Card-vascular/resp. fail.', 'Shock', 'Struck against', 'Inhalation',

        'Fall (same level)', 'Absorption', 'Rubbed/abraded', 'Bite/sting/scratch', 'Rep. Motion/pressure', 'Ingestion']
```

```
frame = pd.read_csv('dataset/test.csv')

frame = frame.dropna()

frame['Event type'] = frame['Event type'].replace('Caught in or between',0)

frame['Event type'] = frame['Event type'].replace('Other',1)

frame['Event type'] = frame['Event type'].replace('Fall (from elevation)',2)

frame['Event type'] = frame['Event type'].replace('Struck-by',3)

frame['Event type'] = frame['Event type'].replace('Card-vascular/resp. fail.',4)

frame['Event type'] = frame['Event type'].replace('Shock',5)

frame['Event type'] = frame['Event type'].replace('Struck against',6)

frame['Event type'] = frame['Event type'].replace('Inhalation',7)

frame['Event type'] = frame['Event type'].replace('Fall (same level)',8)

frame['Event type'] = frame['Event type'].replace('Absorption',9)

frame['Event type'] = frame['Event type'].replace('Rubbed/abraded',10)

frame['Event type'] = frame['Event type'].replace('Bite/sting/scratch',11)

frame['Event type'] = frame['Event type'].replace('Rep. Motion/pressure',12)

frame['Event type'] = frame['Event type'].replace('Ingestion',13)
```

```
def process_text(text):

    nopunc = [char for char in text if char not in string.punctuation]

    nopunc = ''.join(nopunc)

    clean_words = [word for word in nopunc.split() if word.lower() not in stopwords.words('english')]

    return clean_words



cv = CountVectorizer(analyzer=process_text,stop_words = "english", lowercase = True)

messages_bow = cv.fit_transform(frame['Abstract Text'])

X_train, X_test, y_train, y_test = train_test_split(messages_bow, frame['Event type'], test_size = 0.20, random_state = 0)

cpickle.dump(cv.vocabulary_,open("model/feature.pkl","wb"))

cpickle.dump(X_train,open("model/xtrain.pkl","wb"))

cpickle.dump(X_test,open("model/xtest.pkl","wb"))

cpickle.dump(y_train,open("model/ytrain.pkl","wb"))

cpickle.dump(y_test,open("model/ytest.pkl","wb"))

import numpy as np

import pandas as pd

import nltk

from nltk.corpus import stopwords
```

```
import string

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB

import pickle as cpickle

from sklearn import svm

from sklearn.metrics import accuracy_score

msg = 'killed while standing beside a conveyor when he was struck by a fallen'

cvv                                                                    =
CountVectorizer(decode_error="replace",vocabulary=cpickle.load(open("model/feature.pkl",
"rb")))

cv1 = CountVectorizer(vocabulary=cvv.get_feature_names(),stop_words = "english", lowercase
= True)

test = cv1.fit_transform([msg])

X_train = cpickle.load(open("model/xtrain.pkl", "rb"))

X_test = cpickle.load(open("model/xtest.pkl", "rb"))

y_train = cpickle.load(open("model/ytrain.pkl", "rb"))

y_test = cpickle.load(open("model/ytest.pkl", "rb"))

print(X_train.shape)
```

```
def prediction(X_test, cls):  #prediction done here

    y_pred = cls.predict(X_test)

    for i in range(len(X_test)):

        print("X=%s, Predicted=%s" % (X_test[i], y_pred[i]))

    return y_pred

            # Function to calculate accuracy

def cal_accuracy(y_test, y_pred):

    accuracy = accuracy_score(y_test,y_pred)*100

    return accuracy

cls = svm.SVC(kernel='linear', class_weight='balanced', probability=True)

cls.fit(X_train.toarray(), y_train)

prediction_data = prediction(X_test.toarray(), cls)

svm_acc = cal_accuracy(y_test, prediction_data)

print(svm_acc)

cvv = CountVectorizer(decode_error="replace",vocabulary=cpickle.load(open("model/feature.pkl", "rb")))

cv1 = CountVectorizer(vocabulary=cvv.get_feature_names(),stop_words = "english", lowercase = True)

test = cv1.fit_transform([msg])
```

```
print('Predicted value: ',cls.predict(test.toarray()))
```

## 4.2 SOFTWARE TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the
Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

## 4.3 TYPES OF TESTS

### 4.3.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application.it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

### 4.3.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome  of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

### 4.3.3 FUNCTIONAL TEST

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised. Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

### 4.3.4 SYSTEM TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

### 4.3.5 WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

## 4.3.6 BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot –seel into it. The test provides inputs and responds to outputs without considering how the software works.

## 4.3.7 UNIT TESTING:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

### Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

### Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

### Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

## 4.3.8 INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated

software components on a single platform to produce failures caused by interface defects.
The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

## 4.3.9 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

# CHAPTER 5

# CONCLUSION

We proposed an approach to automatically extract valid accident precursors from a dataset of raw construction injury reports. Such information is highly valuable, as it can be used to better understand, predict, and prevent injury occurrence. For each of three supervised models (two of which being deep learning-based), we provided a methodology to identify (after training) the textual patterns that are, on average, the most predictive of each safety outcome. We verified that the learned precursors are valid and made several suggestions to improve the results. The proposed methods can also be used by the user to visualize and understand the models' predictions. Incidentally, while predictive skill is high for all models, we make the interesting observation that the simple TF-IDF + SVM approach is on par with (or outperforms) deep learning most of the time.

# CHAPTER 6

## REFERNCES:

[1] S.J. Bertke, A.R. Meyers, S.J. Wurzelbacher, J. Bell, M.L. Lampl, D. Robins, Development and evaluation of a Naïve Bayesian model for coding causation of workers compensation claims, J.Saf.Res.43(5–6) (2012) 327–332, **https://doi.org/10.1016/j.jsr.2012.10.012**

[2] J.A. Taylor, A.V. Lacovara, G.S. Smith, R. Pandian, M. Lehto, Near-miss narratives from the fire service: a Bayesian analysis, Accid. Anal. Prev. 62 (2014) 119–129, https://doi.org/10.1016/j.aap.2013.09.012.

[3] H.M. Wellman, M.R. Lehto, G.S. Sorock, G.S. Smith, Computerized coding of injury narrative data from the National Health Interview Survey, Accid. Anal. Prev. 36 (2) (2004) 165–171, https://doi.org/10.1016/S0001-4575(02)00146-X.

[4] F. Abdat, S. Leclercq, X. Cuny, C. Tissot, Extracting recurrent scenarios from narrative texts using a Bayesian network: application to serious occupational accidents with movement disturbance, Accid. Anal. Prev. 70 (2014) 155–166, https://doi. org/10.1016/j.aap.2014.04.004.

[5] H.R. Marucci-wellman, H.L. Corns, M.R. Lehto, Classifying injury narratives of large administrative databases for surveillance - a practical approach combining machine learning ensembles and human review, Accid. Anal. Prev. 98 (2017) 359–371, https://doi.org/10.1016/j.aap.2016.10.014.

[6] S.J. Bertke, A.R. Meyers, S.J. Wurzelbacher, A. Measure, M.P. Lampl, D. Robins, Comparison of methods for auto-coding causation of injury narratives, Accid. Anal. Prev. 88 (2016) 117–123, https://doi.org/10.1016/j.aap.2015.12.006.

[7] A.J. Tixier, M.R. Hallowell, B. Rajagopalan, D. Bowman, Application of machine learning to construction injury prediction, Autom. Constr. 69 (2016) 102–114, https://doi.org/10.1016/j.autcon.2016.05.016.

[8] A.J. Tixier, M.R. Hallowell, B. Rajagopalan, D. Bowman, Automation in construction automated content analysis for construction safety: a natural language processing system to extract precursors and outcomes from unstructured injury reports, Autom. Constr. 62 (2016) 45–56, https://doi.org/10.1016/j.autcon.2015.11.001.

[9] Y.M. Goh, C.U. Ubeynarayana, Construction accident narrative classification: an evaluation of text mining techniques, Accid. Anal. Prev. 108 (2017) 122–130, https://doi.org/10.1016/j.aap.2017.08.026. [14] C.U. Ubeynarayana, Y.M. Goh, An Ensemble Approach for Classification of Accident Narratives ASCE International Workshop on Computing in Civil Engineering 2017, (2017), pp. 409–416, https://doi.org/10.1061/9780784480847.051.

[10] A. Chokor, H. Naganathan, W.K. Chong, M. El, Analyzing Arizona OSHA injury reports using unsupervised machine learning, Procedia Eng. 145 (2016) 1588–1593, https://doi.org/10.1016/j.proeng.2016.04.200.

[11] H. Fan, H. Li, Retrieving similar cases for alternative dispute resolution in construction accidents using text mining techniques, Autom. Constr. 34 (2013) 85–91, https://doi.org/10.1016/j.autcon.2012.10.014.

[12] Y. Zou, A. Kiviniemi, S.W. Jones, Retrieving similar cases for construction project risk management using Natural Language Processing techniques, Autom. Constr. 80 19th International Conference on Pattern Recognition, 2008, pp. 1–4, https://ieeexplore.ieee.org/document/4761096