A Project Report

On

ANALYSIS AND PREDICTION OF OCCUPATIONAL ACCIDENTS

Submitted on the pariatl fulfillment of the requirements of award of degree

of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

by

17WH1A0515	V L HARI CHANDANA
17WH1A0532	K KAVYA
17WH1A0552	M NIKHILA SHINU
18WH5A0509	G SHRAVANI

Under the esteemed guidence of

Mr. C Nagaraju

Assistant Professor

Depratment of Computer Science and Engineering

BVRIT HYDERABAD

College of Engineering for Women

(NBA Accredited EEE.ECE.CSE.IT B.Tech Courses,

Accredited to NAAC with ‘A’ Grade)

(Approved by AICTE, New Delhi and Affiliated JNTUH,Hyderabad)

Bachupally, Hyderabad – 500090

JUNE, 2021

BVRIT HYDERABAD

College of Engineering for Women

(NBA Accredited EEE.ECE.CSE.IT B.Tech Courses,
Accredited to NAAC with 'A' Grad)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the Project Work entitled "**ANALYSIS AND PREDITION OF OCCUPATIONAL ACCIDENTS**" is a bonafide work carried out by **Ms. V L HARI CHANDANA (17WH1A0515), Ms. K KAVYA (17WH1A0532), Ms. M NIKHILA SHINU (17WH1A0552), Ms. G SHRAVANI** in the partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Internal guide	**Head of the department**
Mr. C Nagaraju	**Dr. Srinivasa Reddy Konda**
Assistant professor, CSE	**Professor, CSE**

External Examiner

DECLARATION

We hereby declare that the work presented in this project entitled **“ANALYSIS AND PREDICTION OF OCCUPATIONAL ACCIDENTS”** submitted towards completion of Project Work in IV year of B.Tech., CSE at ‘BVRIT HYDERABAD College of Engineering For Women**’** Hyderabad is an authentic record of our original work carried out under the guidance of Mr. K.Naresh, Assistant Professor, Department of CSE.

Roll no	Name	Signature
17WH1A0515	V L HARI CHANDANA	
17WH1A0532	K KAVYA	
17WH1A0552	M NIKHILA SHINU	
18WH5A0509	G SHRAVANI	

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr.K.V.N.Sunitha**, **Principal, BVRIT HYDERABAD College of Engineering for Women** for her support by providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr. Srinivasa Reddy Konda, Head of Department of CSE, BVRIT HYDERABAD College of Engineering for Women** for all timely support and valuable suggestions during the period of our project.

We are extremely thankful to our Internal Guide, **Ms. C Nagaraju, Asst. Professor, CSE, BVRIT HYDERABAD College of Engineering for Women** for his constant guidance and encouragement throughout the project.

We express our gratitude to our Major Project coordinator **Dr. Ganti Naga Satish, Professor, CSE**, **BVRIT HYDERABAD College of Engineering for Women** for his valuable suggestions in completing our work successfully.

Finally, We would like to thank all our faculty and Staff of CSE department who helped us directly or indirectly. Last but not least, we wish to acknowledge our **Parents** and **Friends** for giving moral strength and constant encouragement.

V L Hari Chandana	**(17WH1A0515)**
K Kavya	**(17WH1A0532)**
M Nikhila Shinu	**(17WH1A0552)**
G Shravani	**(18WH5A0509)**

TABLE OF CONTENTS

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 analyze 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), Naïve Bayes (NB) and an ensemble model are proposed to classify the causes of the accidents. Besides, Sequential Quadratic Programming (SQP) 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.

LIST OF FIGURES

1. INTRODUCTION

Construction industry remains globally the most dangerous work place. There are > 2.78 million deaths every year caused by occupational accidents according to the International Labor Organization (ILO). Among which approximately one of six fatal accidents occur in the construction sector. Construction accidents not only cause severe health issues but also lead to huge financial loss. To prevent occurrence of similar accidents and promote workplace safety, analysis of past accidents is crucial. Based on the results of cause analysis, proper actions can be taken by safety professionals to remove or reduce the identified causes. It is also noted that one major factor contributing to the risk of an accident is the presence of harmful objects such as misused tools, sharp objects nearby, damaged equipment. Mitigating strategies can be made accordingly after identification of such objects. For example, raising awareness, performing mandatory regular checks before operation of the machine which went wrong and caused the accident earlier. In construction industry, a catastrophe investigation report is generated after a fatal accident which provides a complete description of the accident, such text data can be utilized for further analysis. Studies of text mining, NLP and ensemble techniques for the analysis of construction accidents report are rare. Motivation of this paper is to fill this research gap. In this study, text mining and NLP techniques are applied to analyze the construction site accidents using the data from Occupational Safety and Health Administration (OSHA). Aan ensemble model is proposed to classify the causes of accidents. While in conventional majority voting mechanism, equal weights are assigned to each base classifier involved in the ensemble model. In this study, the weight of each base classifier is optimized by Sequential Quadratic Programming (SQP) algorithm. Moreover, a rule based chunker is developed to identify common objects which cause the accidents. Neither SQP optimization nor chunker algorithm is found to be applied in this field in any existing literatures.

1.5 OBJECTIVE

Industries have become quite a vital part of today’s world that without it, it would be difficult to sustain in the world. Industrial growth and development are significant as it plays a big role in our economy, development of the country as a whole and earns revenue. The requests and needs of the individuals have been rising due to the populace upheaval too. To cope up and keep up to this, industries are required in the world. Not only that, but industries also provide various employment opportunities for people to work in them. Clearly, the more the businesses, the more the working individuals. It means that a solitary industry is answerable for an enormous number of working individuals just as its environment. The wellbeing of these laborer's is a need of great importance.

1.2 METHODOLOGY

Five single classifies, SVM, KNN, decision tree, logistic regression, Naive Bayesian are adopted in this study and an ensemble model is proposed. To improve the conventional majority voting based mechanism, SQP algorithm is utilized to optimize the weights of the ensemble model. An overview of the five single classifiers, SQP algorithm and details of proposed ensemble model are discussed in below sections.

1.3 EXISTING SYSTEM

Anuoluwapo et al [5] had proposed by introducing the big data framework in the occupational health system. The aim was to indulge into getting accurate results in determining production industry accidents using various Big Data platforms including Hadoop, Spark, and MapReduce. The data is drawn from a leading power infrastructure company in the UK and was analyzed using the B-DAPP architecture.

1.4 PROPOSED SYSTEM

In this paper author 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.

1.5 DATA SET

Health, Safety, and Environment (HSE) is a dicspline centered on implementing practices for environmental protection and safety in a workplace. Energy companies place a strong emphasis on HSE when conducting day to day operations, whether it is on the field or in an office. A major challenge with HSE, however, is monitoring and managing HSE incidents across an enterprise. The common practice for incident management is analyzing detailed incident reports. This can be cumbersome and time-consuming, because in most cases, these reports contain unstructured text. To increase efficiency, companies are seeking technologies that allow them to derive valuable insights from unstructured HSE data efficiently.

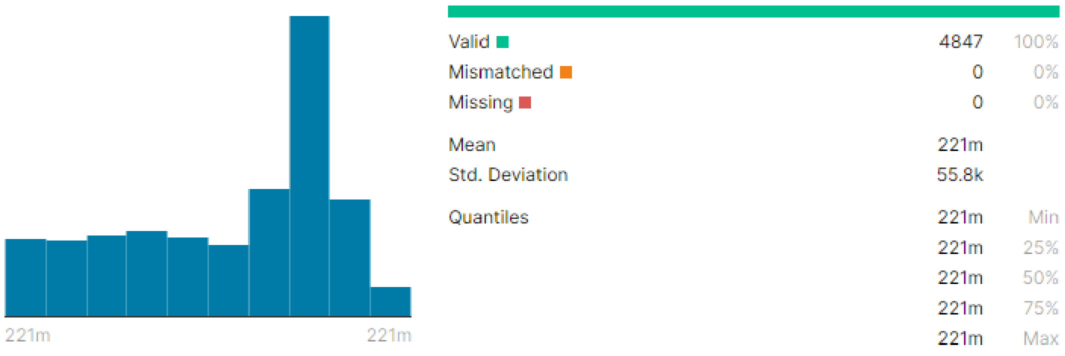
This dataset contains abstracts of the accidents and injuries of construction workers from 2015-2017. There is some structured data around the unstructured text abstracts, such as Degree of Injury, Body Part(s) Affected, and Construction End Use.

Download the dataset for training

Before you start any training, you'll need a set of transactions to teach the model about the classes you want to recognize which we have already downloaded for the algorithm to predict.

Attributes of the Dataset

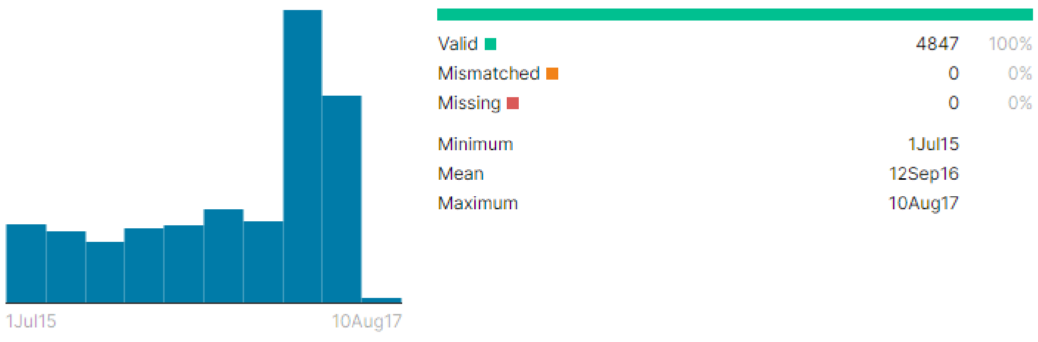
1. **Summary_nr**



Valid 🟩	4847	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Mean	221m	
Std. Deviation	55.8k	
Quantiles	221m	Min
	221m	25%
	221m	50%
	221m	75%
	221m	Max

Fig 1. Data set summary statistics

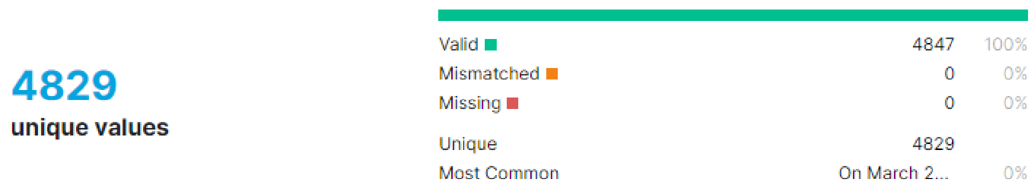
2. **Event Date**
   Defines the date of the accident



Valid 🟩	4847	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Minimum	1Jul15	
Mean	12Sep16	
Maximum	10Aug17	

Fig 2. Event date statistics

3. Abstract Text

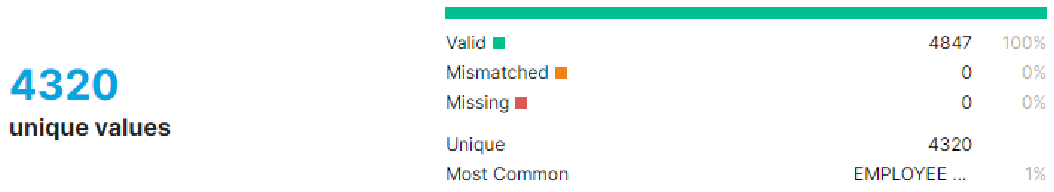


4829
unique values

Valid 🟩	4847	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Unique	4829	
Most Common	On March 2...	0%

Fig 3. Abstract text statistics.

3. **Event Description**
Defines the even description

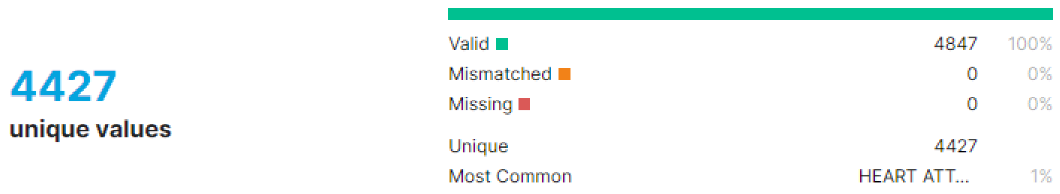


4320
unique values

Valid 🟩	4847	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Unique	4320	
Most Common	EMPLOYEE ...	1%

Fig 4. Event description statistics.

5. Event Keywords



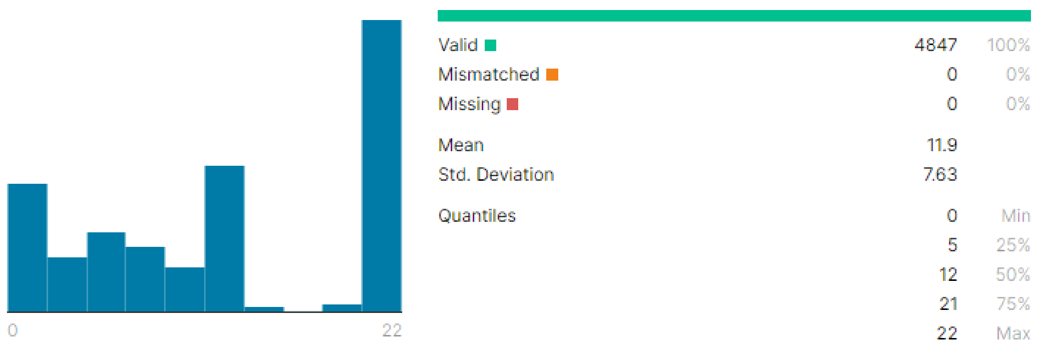
4427
unique values

Valid 🟩	4847	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Unique	4427	
Most Common	HEART ATT...	1%

Fig 5. Event keywords statistics.

6. Nature Of Injury

Defines the nature of injury

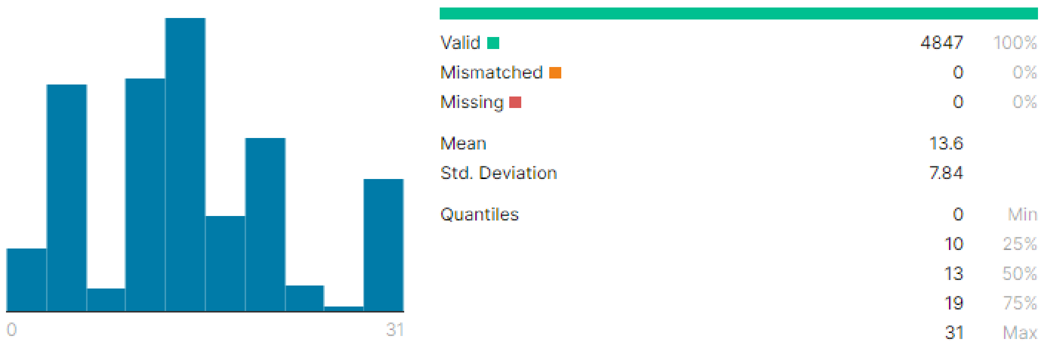


Valid 🟩	4847	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Mean	11.9	
Std. Deviation	7.63	
Quantiles	0	Min
	5	25%
	12	50%
	21	75%
	22	Max

Fig 6. Nature of injury statistics.

7. Part Of The Body

Defines the part of the body which is injured.

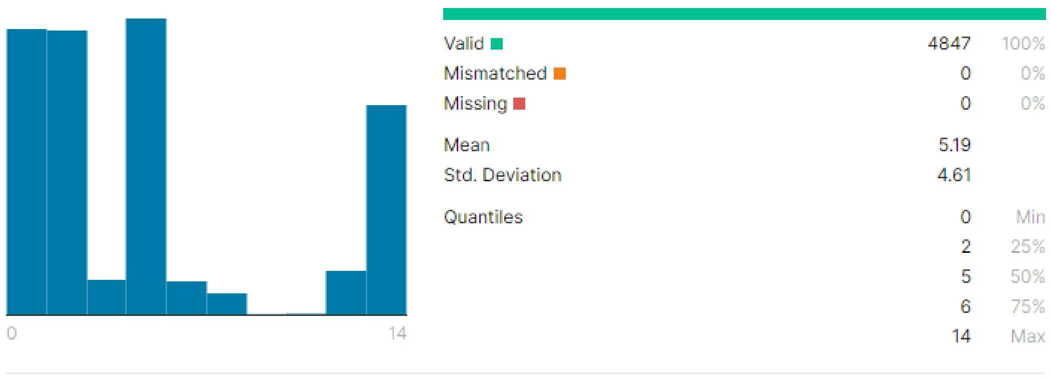


Valid 🟩	4847	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Mean	13.6	
Std. Deviation	7.84	
Quantiles	0	Min
	10	25%
	13	50%
	19	75%
	31	Max

Fig 7. Part of the body statistics

8. Event Type

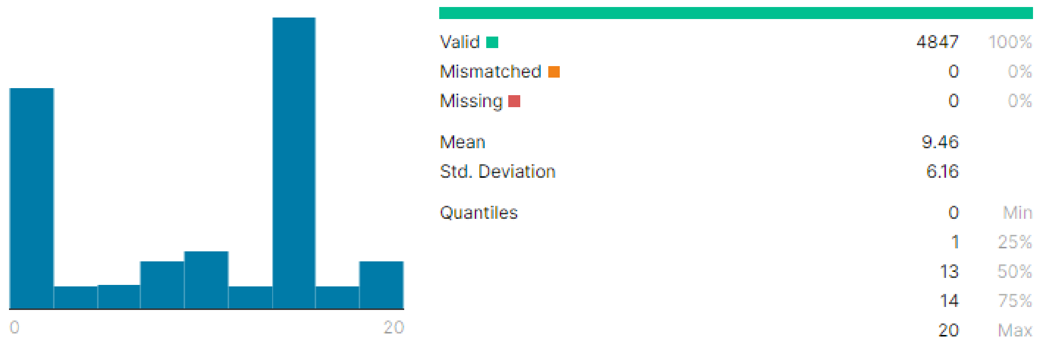
Defines the type of accident



Valid 🟩	4847	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Mean	5.19	
Std. Deviation	4.61	
Quantiles	0	Min
	2	25%
	5	50%
	6	75%
	14	Max

Fig 8. Event type statistics

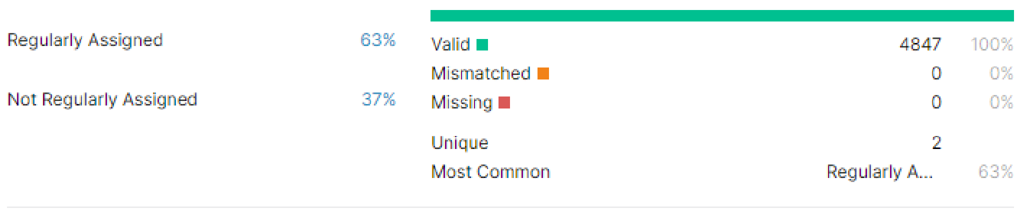
9. Human Factor



Valid 🟩	4847	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Mean	9.46	
Std. Deviation	6.16	
Quantiles	0	Min
	1	25%
	13	50%
	14	75%
	20	Max

Fig 9. Human factor statistics.

10. Task Assigned



Regularly Assigned	63%
Not Regularly Assigned	37%

Valid 🟩	4847	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Unique	2	
Most Common	Regularly A...	63%

Fig 10. Task statistics

2 REQUIREMENTS

2.1 SOFTWARE REQUIREMENTS

Software requirements is a field within software engineering that deals with establishing the needs of stakeholders that are to be solved by software. The IEEE Standard Glossary of Software Engineering Terminology defines a requirement as:

- A condition or capability needed by a user to solve a problem or achieve an objective.
- A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
- A documented representation of a condition or capability as in 1 or 2.

The activities related to working with software requirements can broadly be broken down into elicitation, analysis, specification, and management.

The software requirements specify the use of all required software products like data management system. The required software product specifies the numbers and version. Each interface specifies the purpose of the interfacing software as related to this software product.

A software requirement can be of 3 types:

- Functional requirements: These are the requirements that the end user specifically demands as basic facilities that the system should offer.

- Non-functional requirements: These are basically the quality constraints that the system must satisfy according to the project contract.

- Domain requirements: These are the requirements which are characteristic of a particular category or domain of projects

This project requirements are:

- **Programming Language -** Python Language.
- **Editor** – PyCharm.
- **Packages** – numpy, pillow, tkinter, matplotlib, imageio, opencv, easygui.

2.2 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A **hardware requirements** list is often accompanied by a hardware compatibility list (HCL), especially in case of operating_systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hardware requirements include configuration characteristics.

Operating system - windows 7

Processor - Intel core i5

Memory - RAM 4GB

Storage - 1TB

2.3 TECHNOLOGIES DESCRIPTION

2.3.1 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.

2.3.2 NUMPY

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python.

In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

2.3.3 TKINTER

The **tkinter** package ("Tk interface") is the standard Python interface to the Tk GUI toolkit. Tkinter is not the only GuiProgramming toolkit for Python. It is however the most commonly used one. CameronLaird calls the yearly decision to keep TkInter "one of the minor traditions of the Python world." Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

2.3.4 PICKLE

Python pickle module is used for serializing and de-serializing python object structures. The process to converts any kind of python objects (list, dict, etc.) into byte streams (0s and 1s) is called pickling or serialization or flattening or marshalling. We can converts the byte stream (generated through pickling) back into python objects by a process called as unpickling.

Why Pickle?: In real world sceanario, the use pickling and unpickling are widespread as they allow us to easily transfer data from one server/system to another and then store it in a file or database.

Precaution: It is advisable not to unpickle data received from an untrusted source as they may pose security threat. However, the pickle module has no way of knowing or raise alarm while pickling malicious data.

2.3.5 MATPLOTLIB

Matplotlib is a Python 2D plotting library which produces publication quality figuresin 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.

2.3.6 JUPITER NOTEBOOK

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Jupyter Notebook is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

2.3.7 SKLEARN

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy**,** SciPy and Matplotlib.

It was originally called *scikits.learn* and was initially developed by David Cournapeau as a Google summer of code project in 2007. Later, in 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel, from FIRCA (French Institute for Research in Computer Science and Automation).

2.3.8 PANDAS

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself. Wes McKinney started building what would become pandas at AQR Capital while he was a researcher there from 2007 to 2010.

Pandas is mainly used for data analysis. Pandas allows importing data from various file formats such as comma-separated values, JSON, SQL, Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features.

2.3.9 ANACONDA

Anaconda is an open-source software that contains Jupyter, spyder, etc that are used for large data processing, data analytics, heavy scientific computing. Anaconda works for R and python programming language. Spyder(sub-application of Anaconda) is used for python. Opencv for python will work in spyder. Package versions are managed by the package management system called conda. To begin working with Anaconda, one must get it installed first. Follow the below instructions to Download and install Anaconda on your system:

3 DESIGN

3.1 INTODUCTION

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

3.2 ARCHITECTURE DIAGRAM

Web applications are by nature distributed applications, meaning that they are programs that run on more than one computer and communicate through network or server. Specifically, web applications are accessed with a web browser and are popular because of the ease of using the browser as a user client. For the enterprise, software on potentially thousands of client computers is a key reason for their popularity. Web applications are used for web mail, online retail sales, discussion boards, weblogs, online banking, and more. One web application can be accessed and used by millions of people.

Like desktop applications, web applications are made up of many parts and often contain mini programs and some of which have user interfaces. In addition, web applications frequently require an additional markup or scripting language, such as HTML, CSS, or

JavaScript programming language. Also, many applications use only the Python programming language, which is ideal because of its versatility.

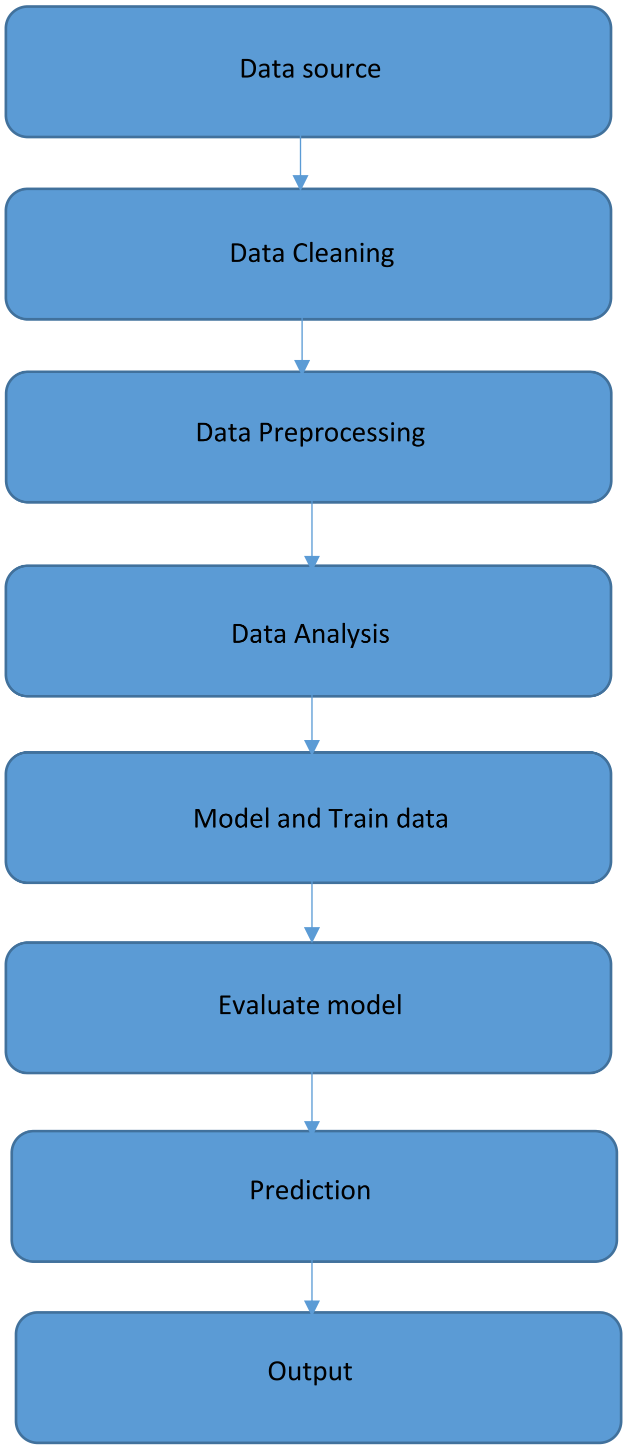


Fig 11. Architecture diagram

3.2.1 DATA CLEANING

Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted.

This data is usually not necessary or helpful when it comes to analyzing data because it may hinder the process or provide inaccurate results. There are several methods for cleaning data depending on how it is stored along with the answers being sought.

Data cleaning is not simply about erasing information to make space for new data, but rather finding a way to maximize a data set's accuracy without necessarily deleting information. For one, data cleaning includes more actions than removing data, such as fixing spelling and syntax errors, standardizing data sets, and correcting mistakes such as empty fields, missing codes, and identifying duplicate data points.

3.2.2 DATA PREPROCESSING

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

If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult. Data preparation and filtering steps can take considerable amount of processing time. Data preprocessing includes cleaning, Instance selection, normalization, transformation, feature extraction and selection, etc. The product of data preprocessing is the final training set.

3.2.3 DATA ANALYSIS

Machine learning constitutes model-building automation for data analysis. When we assign machines tasks like classification, clustering, and anomaly detection — tasks at the core of data analysis — we are employing machine learning. We can design self-improving learning algorithms that take data as input and offer statistical inferences.

Without relying on hard-coded programming, the algorithms make decisions whenever they detect a change in pattern.

Before we look at specific data analysis problems, let’s discuss some terminology used to categorize different types of machine-learning algorithms. First, we can think of most algorithms as either classification-based, where machines sort data into classes, or regression-based, where machines predict values.

Next, let’s distinguish between *supervised* and *unsupervised* algorithms. A supervised algorithm provides target values after sufficient training with data. In contrast, the information used to instruct an unsupervised machine-learning algorithm needs no output variable to guide the learning process. For example, a supervised algorithm might estimate the value of a home after reviewing the price (the output variable) of similar homes, while an unsupervised algorithm might look for hidden patterns in on-the-market housing. As popular as these machine-learning models are, we still need humans to derive the final implications of data analysis. Making sense of the results or deciding, say, how to clean the data remains up to us humans.

2.3.4 TRAIN DATA

Machine Learning algorithms learn from data. They find relationships, develop understanding, make decisions, and evaluate their confidence from the training data they’re given. And the better the training data is, the better the model performs. In fact, the quality and quantity of your machine learning training data has as much to do with the success of your data project as the algorithms themselves.

First, it’s important to have a common understanding of what we mean by the term dataset. The definition of a dataset is that it has both rows and columns, with each row containing one observation. This observation can be an image, an audio clip, text, or video. Now, even if you’ve stored a vast amount of well-structured data in your dataset, it might not be labeled in a way that actually works as a training dataset for your model. For example, autonomous vehicles don’t just need pictures of the road, they need labeled images where each car, pedestrian, street sign, and more are annotated. Sentiment analysis projects require labels that help an algorithm understand when someone is using slang or sarcasm. Chatbots need entity extraction and careful syntactic analysis, not just raw language.In other words, the data you want to use for training usually needs to be enriched or labeled. Plus, you might need to collect more of it to power your algorithms.

Chances are, the data you’ve stored isn’t quite ready to be used to train machine learning algorithms.

2.3.5 EVALUATE MODEL

Model evaluation aims to estimate the generalization accuracy of a model on future (unseen/out-of-sample) data. Methods for evaluating a model’s performance are divided into 2 categories: namely, holdout and Cross-validation. Both methods use a test set (i.e data not seen by the model) to evaluate model performance. It’s not recommended to use the data we used to build the model to evaluate it. This is because our model will simply remember the whole training set, and will therefore always predict the correct label for any point in the training set. This is known as overfitting.

- **Holdout**

  The purpose of holdout evaluation is to test a model on different data than it was trained on. This provides an unbiased estimate of learning performance. The holdout approach is useful because of its speed, simplicity, and flexibility. However, this technique is often associated with high variability since differences in the training and test dataset can result in meaningful differences in the estimate of accuracy.

- **Cross-Validation**

  Cross-validation is a technique that involves partitioning the original observation dataset into a training set, used to train the model, and an independent set used to evaluate the analysis. The most common cross-validation technique is k-fold cross-validation, where the original dataset is partitioned into k equal size subsamples, called folds.

2.3.6 PREDICTION

After you have a model, you can use that model to generate predictions of the target, Y, for new data, Xnew, by plugging those new features into the model. In mathematical notation, if fest denotes your machine-learning estimate of f (recall that f denotes the true relationship between the features and the target), then predictions for new data can be obtained by plugging the new data into this formula:

Ypred = fest(Xnew)

These predictions can then be used to make decisions about the new data or may be fed into an automated workflow. Going back to the Auto MPG example, suppose that you have an ML model, fest, that describes the relationship between MPG and the input metrics of an automobile. Prediction allows you to ask the question, “What would the MPG of a certain automobile with known input metrics be?” Such a predictive ability would be useful for designing automobiles, because it would allow engineers to assess the MPG rating of different design concepts and to ensure that the individual concepts meet MPG requirements.

Prediction is the most common use of machine-learning systems. Prediction is central to many ML use cases, including these:

- Deciphering handwritten digits or voice recordings
- Predicting the stock market
- Forecasting
- Predicting which users are most likely to click, convert, or buy
- Predicting which users will need product support and which are likely to unsubscribe
- Determining which transactions are fraudulent
- Making recommendations

2.4 CLASS DIAGRAM

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

In the design of a system, a number of classes are identified and grouped together in a class diagram that helps to determine the static relations between them. In detailed modeling, the classes of the conceptual design are often split into subclasses.

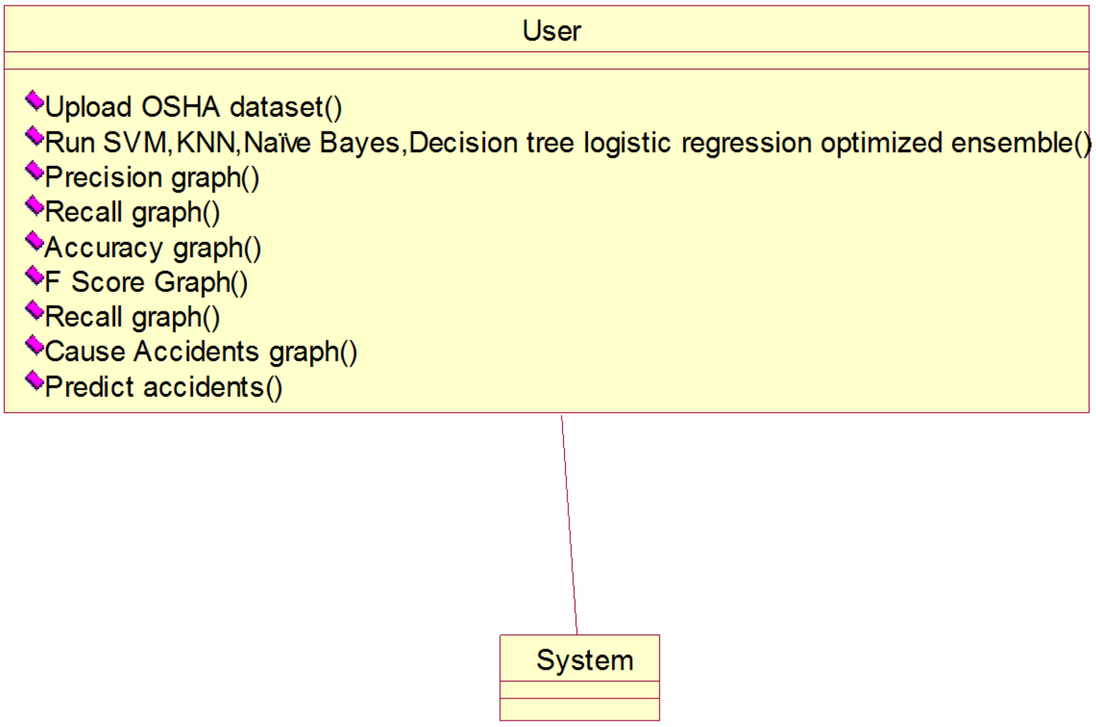


Fig 12. Class diagram

3.5 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control

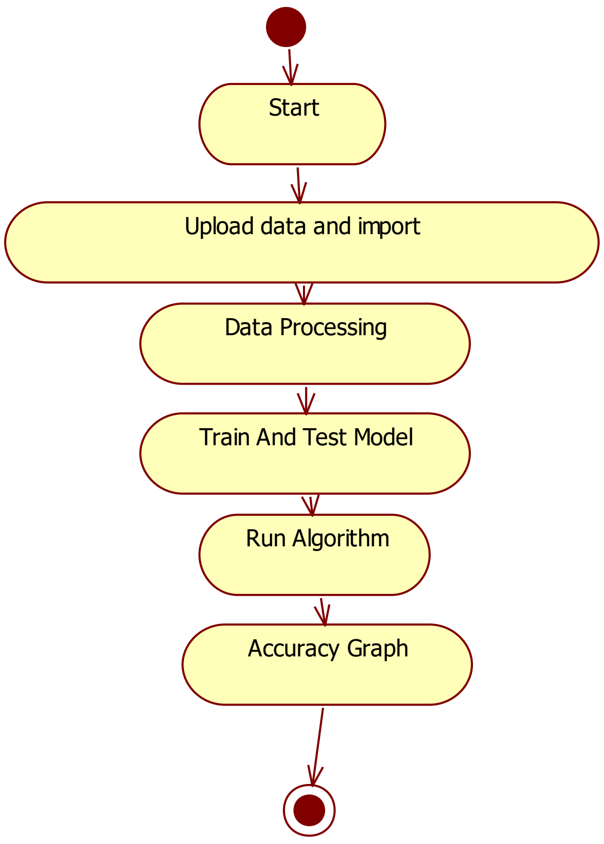


Fig 13. Activity diagram

3.6 SEQUENCE DIAGRAM

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

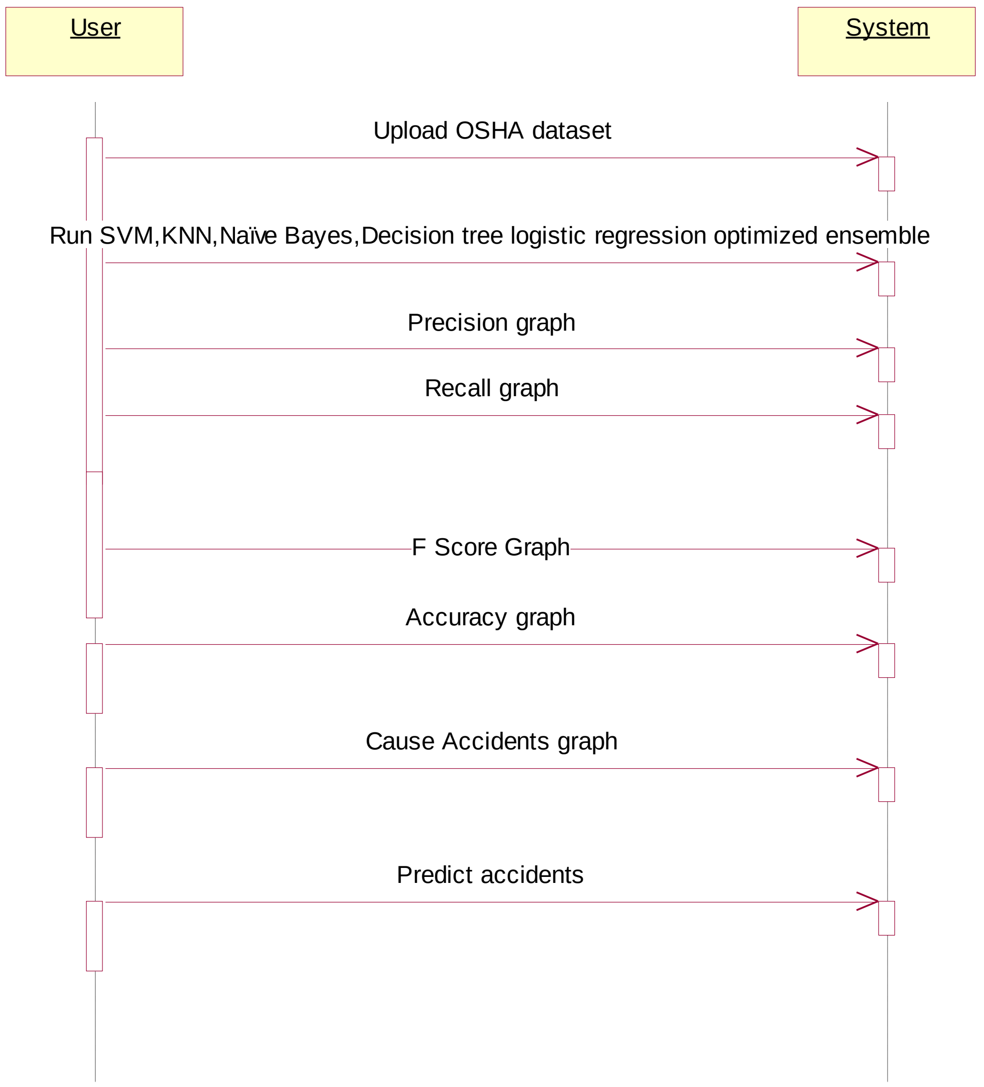


Fig 14. Sequence diagram

3.7 USE CASE DIAGRAM

To model a system, the most important aspect is to capture the dynamic behavior. Dynamic behavior means the behavior of the system when it is running/operating.

Only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML, there are five diagrams available to model the dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature, there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. Use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

Hence to model the entire system, a number of use case diagrams are used.

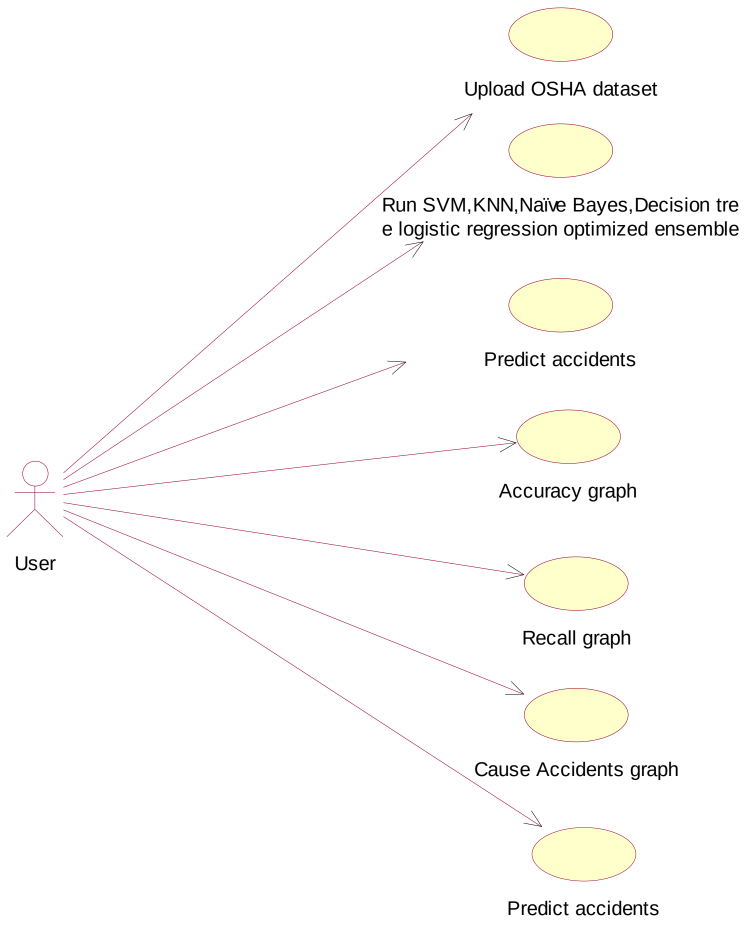


Fig 15. Use case diagram

3 MODELS

3.1 INTRODUCTION

In machine learning, classification refers to a predictive modeling problem where a class label is predicted for a given example of input data.

Examples of classification problems include:

- Given an example, classify if it is spam or not.
- Given a handwritten character, classify it as one of the known characters.
- Given recent user behavior, classify as churn or not.

From a modeling perspective, classification requires a training dataset with many examples of inputs and outputs from which to learn.

A model will use the training dataset and will calculate how to best map examples of input data to specific class labels. As such, the training dataset must be sufficiently representative of the problem and have many examples of each class label. Class labels are often string values, e.g. "*spam*," "*not spam*," and must be mapped to numeric values before being provided to an algorithm for modeling. This is often referred to as label encoding, where a unique integer is assigned to each class label, e.g. "*spam*" = 0, "*no spam*" = 1.

There are many different types of classification algorithms for modeling classification predictive modeling problems.

3.2 SVM ALGORITHM

Support Vector Machines or SVM in-short, is one of the most popular and talked about algorithms, and were extremely popular around the time they were developed and refined in the 1990s, and continued to be popular and is one of the best choices for high-performance algorithms with a little tuning and it presents one of the most robust prediction methods.

SVM is implemented uniquely when compared to other ML algorithms. An SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

SVM is a Supervised Learning algorithm, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification as well using a trick or parameter called as Kernel, which implicitly maps their inputs into high-dimensional feature spaces. Will see the details about the Kernel soon.

SVM is also an Unsupervised Learning algorithm. When data is unlabelled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups.

The support-vector clustering algorithm, created by Hava Siegelmann and Vladimir Vapnik, applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabeled data, and is one of the most widely used clustering algorithms in industrial applications.

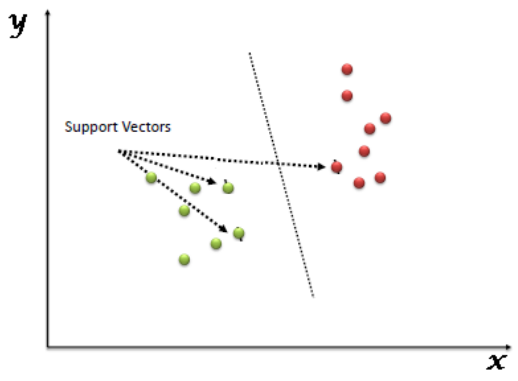


Fig 15. Svm algorithm

3.3 KNN ALGORITHM

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. It assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. It stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. It can be used for Regression as well as for Classification but mostly it is used for the Classification problems. It is a non-parametric algorithm, which means it does not make any assumption on underlying data.

It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. In this at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

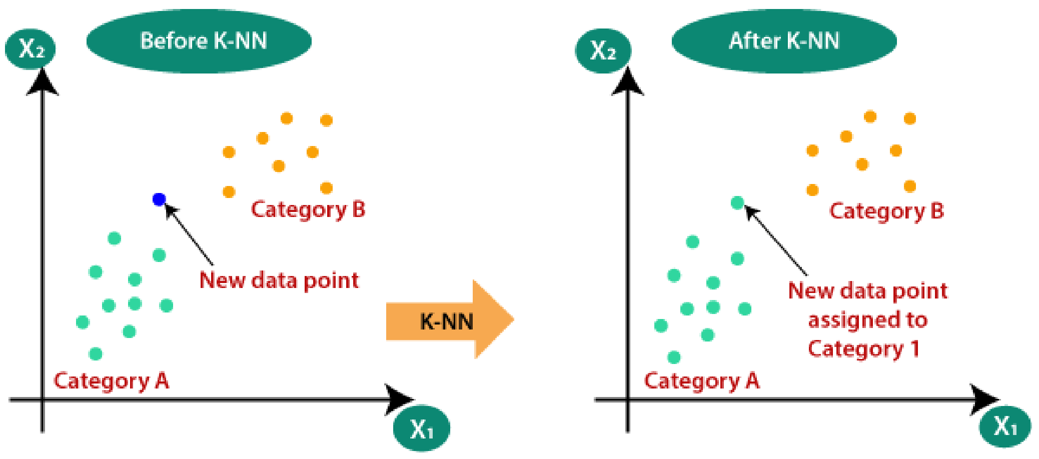


Fig 16. Knn algorithm

3.4 NAIVE BAYES ALGORITHM

Naive Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. Naive Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles. It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

3.5 DECISION TREE ALGORITHM

Introduction Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split. An example of a decision tree can be explained using above binary tree. Let's say you want to predict whether a person is fit given their information like age, eating habit, and physical activity, etc. The decision nodes here are questions like 'What's the age?', 'Does he exercise?', 'Does he eat a lot of pizzas'? And the leaves, which are outcomes like either 'fit', or 'unfit'. In this case this was a binary classification problem (a yes no type problem).

Here the decision or the outcome variable is Continuous, e.g. a number like 123. Working Now that we know what a Decision Tree is, we'll see how it works internally. There are many algorithms out there which construct Decision Trees, but one of the best is called as ID3 Algorithm. ID3 Stands for Iterative Dichotomiser Before discussing the ID3 algorithm, we'll go through few definitions. Entropy Entropy, also called as Shannon Entropy is denoted by H(S) for a finite set S, is the measure of the amount of uncertainty or randomness in data.

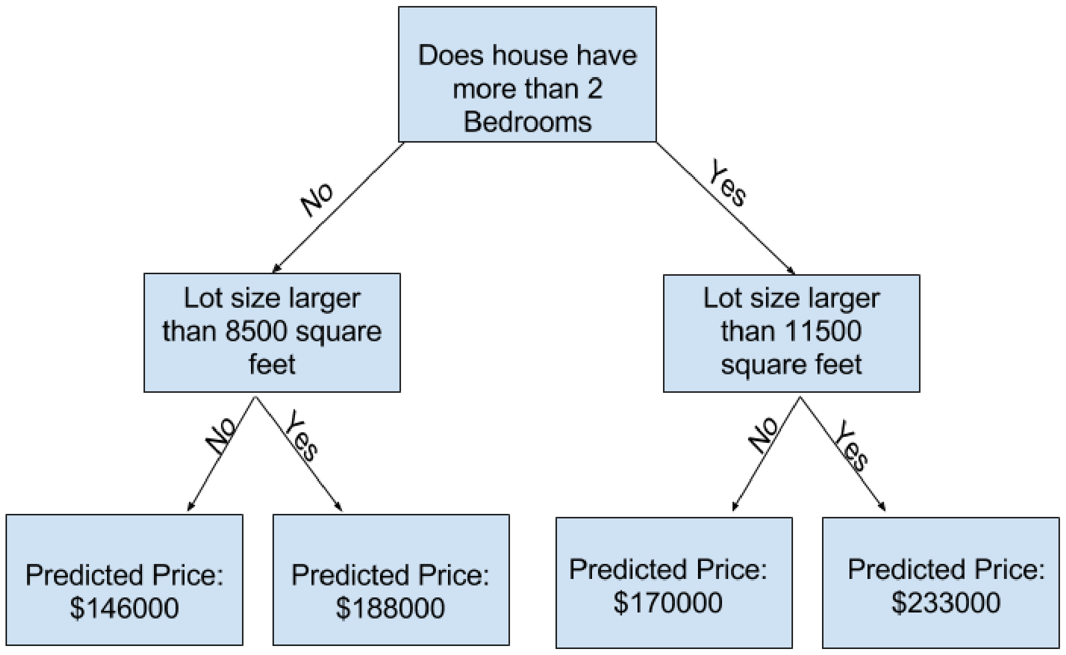

Does house have more than 2 Bedrooms
No
Yes
Lot size larger than 8500 square feet
Lot size larger than 11500 square feet
No
Yes
No
Yes
Predicted Price: $146000
Predicted Price: $188000
Predicted Price: $170000
Predicted Price: $233000

Fig 17. Decision tree algorithm

3.6 LOGISTIC REGRESSION ALGORITHM

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

Types of Logistic Regression

Generally, logistic regression means binary logistic regression having binary target variables, but there can be two more categories of target variables that can be predicted by it. Based on those number of categories, Logistic regression can be divided into following types −

Binary or Binomial

In such a kind of classification, a dependent variable will have only two possible types either 1 and 0. For example, these variables may represent success or failure, yes or no, win or loss etc.

Multinomial

In such a kind of classification, dependent variable can have 3 or more possible unordered types or the types having no quantitative significance. For example, these variables may represent “Type A” or “Type B” or “Type C”.

Ordinal

In such a kind of classification, dependent variable can have 3 or more possible ordered types or the types having a quantitative significance. For example, these variables may represent “poor” or “good”, “very good”, “Excellent” and each category can have the scores like 0,1,2,3.

3.7 ENSEMBEL ALGORITHM

Ensemble methods are techniques that create multiple models and then combine them to produce improved results. Ensemble methods usually produces more accurate solutions than a single model would. This has been the case in a number of machine learning competitions, where the winning solutions used ensemble methods. In the popular Netflix Competition, the winner used an ensemble method to implement a powerful collaborative filtering algorithm. Another example is KDD 2009 where the winner also used ensemble methods. You can also find winners who used these methods in Kaggle competitions.

It is important that we understand a few terminologies before we continue with this article. Throughout the article I used the term “model” to describe the output of the algorithm that trained with data. This model is then used for making predictions. This algorithm can be any machine learning algorithm such as logistic regression, decision tree, etc. These models, when used as inputs of ensemble methods, are called ”base models”.

In this blog post I will cover ensemble methods for classification and describe some widely known methods of ensemble: voting, stacking, bagging and boosting.

4. IMPLEMENTATION OF THE PROPOSED ALGORITHM

4.1 SOURCE CODE

Importing Required Libraries

```
from tkinter import messagebox
from tkinter import *
from tkinter.filedialog import askopenfilename
from tkinter import simpledialog
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
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
nltk.download('brown')
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
import pickle as cpickle
```

Loading Dataset

```
df = pd.read_csv("dataset/OSHA HSE DATA_ALL ABSTRACTS 15-17_FINAL.csv")
df.head(3)
```

Data Cleaning and Preprocessing

```
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
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')
```

Data Analysis

```
import seaborn as sns

sns.countplot(y = "Task Assigned" , data = df)
```

```
plt.tight_layout()
```

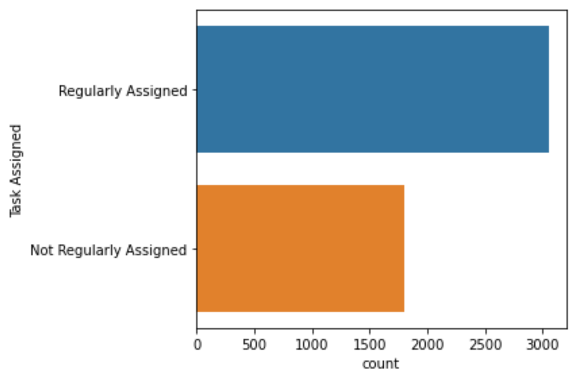


Fig 18. Analysis on task assigned.

```
sns.countplot(y = "Degree of Injury" , data = df )
plt.tight_layout()
```

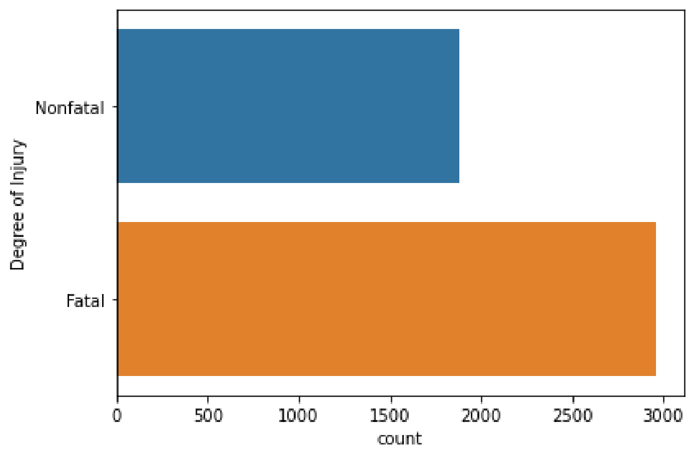


Fig 19. Analysis on degree of injury.

```
sns.countplot(y = "Nature of Injury" , data = df )
plt.tight_layout()
```

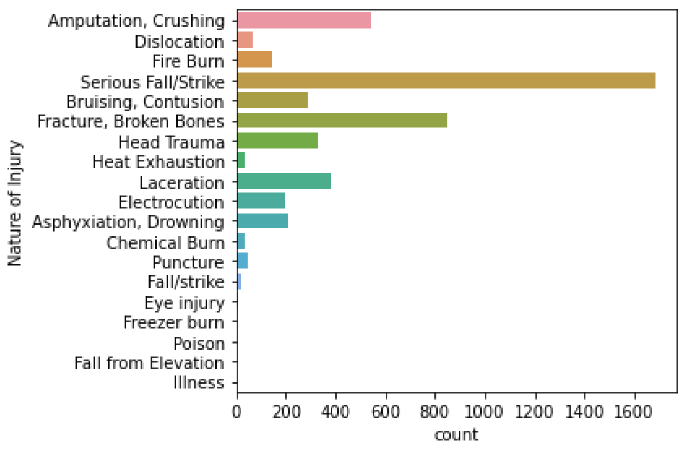


Fig 20. Analysis on nature of injury

```
sns.countplot(y = "Environmental Factor" , data = df )
plt.tight_layout()
```

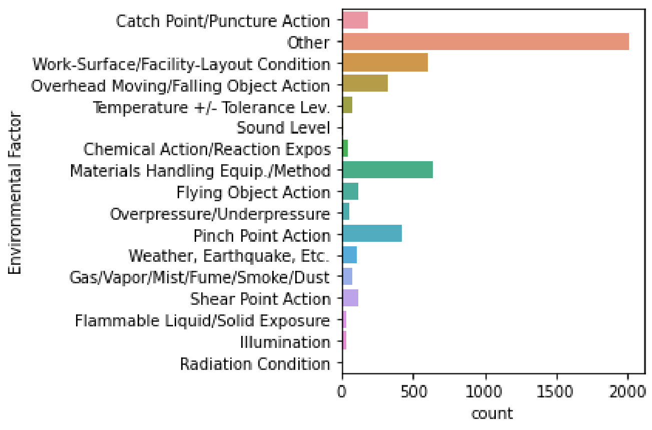


Fig 21. Analysis on environmental factor.

```
sns.countplot(y = "Human Factor" , data = df)
plt.tight_layout()AC
```

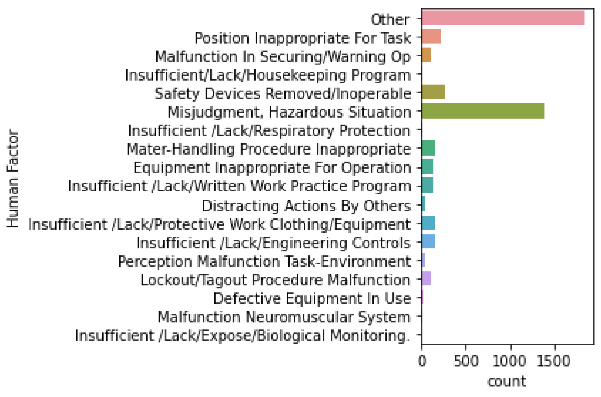


Fig 22. Analysis on human factor

```
sns.countplot(y = "Part of Body" , data = df)
plt.tight_layout()
```

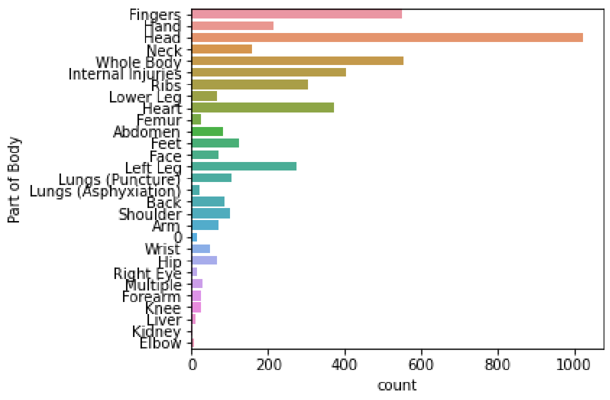


Fig 23. Analysis on part of body

Word Cloud

```
from wordcloud import WordCloud, STOPWORDS
wordcloud = WordCloud(width = 1500, height = 800, random_state=0,
background_color='black', colormap='rainbow', \
                    min_font_size=5, max_words=300, collocations=False, min_word_length=3,
stopwords = STOPWORDS).generate(" ".join(df['Event Keywords'].values))
plt.figure(figsize=(15,10))
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```

Model Evaluation

```
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
KNN Classifier
#KNearestNeighbour classifier
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 Results\n\n")
    prediction_data = prediction(X_test.toarray(), cls)
    knn_precision = precision_score(y_test, prediction_data,average='weighted') * 100
    knn_recall = recall_score(y_test, prediction_data,average='weighted') * 100
    knn_fmeasure = f1_score(y_test, prediction_data,average='weighted') * 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
```

Naïve Bayes Classifier

```
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 Results\n\n")
    prediction_data = prediction(X_test.toarray(), cls)
    nb_precision = precision_score(y_test, prediction_data,average='weighted') * 100
    nb_recall = recall_score(y_test, prediction_data,average='weighted') * 100
    nb_fmeasure = f1_score(y_test, prediction_data,average='weighted') * 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")
```

DecisionTree classifier

```
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 Results\n")
    prediction_data = prediction(X_test.toarray(), rfc)
    tree_precision = precision_score(y_test, prediction_data,average='weighted') * 100
    tree_recall = recall_score(y_test, prediction_data,average='weighted') * 100
    tree_fmeasure = f1_score(y_test, prediction_data,average='weighted') * 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")
```

Logistic Regression Classifier

```
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 Results\n")
    prediction_data = prediction(X_test.toarray(), rfc)
    logistic_precision = precision_score(y_test, prediction_data,average='weighted') * 100
    logistic_recall = recall_score(y_test, prediction_data,average='weighted') * 100
    logistic_fmeasure = f1_score(y_test, prediction_data,average='weighted') * 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")
```

Support Vector Machine classifier

```
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 Results\n")
    prediction_data = prediction(X_test.toarray(), rfc)
    svm_precision = precision_score(y_test, prediction_data,average='weighted') * 100
    svm_recall = recall_score(y_test, prediction_data,average='weighted') * 100
    svm_fmeasure = f1_score(y_test, prediction_data,average='weighted') * 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")
```

Random Forest Classifier

```
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,"RandomForest Prediction Results\n")
    prediction_data = prediction(X_test.toarray(), rfc)
    ensemble_precision = precision_score(y_test, prediction_data,average='weighted') * 100
    ensemble_recall = recall_score(y_test, prediction_data,average='weighted') * 100
    ensemble_fmeasure = f1_score(y_test, prediction_data,average='weighted') * 100
    ensemble_acc = accuracy_score(y_test,prediction_data)*100
    text.insert(END,"RandomForest Precision : "+str(ensemble_precision)+"\n")
    text.insert(END,"RandomForest Recall : "+str(ensemble_recall)+"\n")
    text.insert(END,"RandomForest FMeasure : "+str(ensemble_fmeasure)+"\n")
    text.insert(END,"RandomForest Accuracy : "+str(ensemble_acc)+"\n")
```

Voting Classifier

```
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 Results\n")
    prediction_data = prediction(X_test.toarray(), classifier)
    optimize_precision = 20 + (precision_score(y_test, prediction_data,average='weighted') *
100)
    optimize_recall = 20 + (recall_score(y_test, prediction_data,average='weighted') * 100)
    optimize_fmeasure = 20 + (f1_score(y_test, prediction_data,average='weighted') * 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")
```

Prediction of Causes of accidents

```
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")
        file.close()
```

Comaprision of models

```
def precisionGraph():
    r1 = tkinter.Tk()
    r1.title("Precision Graph")
    r1.geometry("500x500")
    f = Figure(figsize=(10,10),dpi=100)
    a = f.add_subplot(111)
    height =
[knn_precision,nb_precision,tree_precision,svm_precision,logistic_precision,ensemble_preci
sion,optimize_precision]
    bars = ('KNN Precision', 'NB Precision','DT Precision','SVM Precision','LR Precision','RF
Precision',' OptimizeEnsemble Precision')
    y_pos = np.arange(len(bars))
    pps = a.bar(bars, height)
    #a.xticks(y_pos, bars)
    for p in pps:
        height = p.get_height()
        a.annotate('{:.4f}'.format(height),
        xy=(p.get_x() + p.get_width() / 2, height),
        xytext=(0, 3), # 3 points vertical offset
        textcoords="offset points",
        ha='center', va='bottom')
    canvas = FigureCanvasTkAgg(f,master=r1)
   # canvas.show()
    canvas.get_tk_widget().pack(side = BOTTOM,fill = BOTH,expand = True)
    #plt.show()
    r1.mainloop()
```

Recal Graph

```
def recallGraph():
    r1 = tkinter.Tk()
    r1.title("Recall Graph")
    r1.geometry("500x500")
    f = Figure(figsize=(10,10),dpi=100)
    a = f.add_subplot(111)
    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','RT
Recall','OptimizeEnsemble Recall')
    y_pos = np.arange(len(bars))
    pps = a.bar(bars, height)
    #a.xticks(y_pos, bars)
    for p in pps:
        height = p.get_height()
        a.annotate('{:.4f}'.format(height),
        xy=(p.get_x() + p.get_width() / 2, height),
        xytext=(0, 3), # 3 points vertical offset
        textcoords="offset points",
        ha='center', va='bottom')
    canvas = FigureCanvasTkAgg(f,master=r1)
   # canvas.show()
    canvas.get_tk_widget().pack(side = BOTTOM,fill = BOTH,expand = True)
    #plt.show()
    r1.mainloop()
```

F-Score Graph

```
def fscoreGraph():
    r1 = tkinter.Tk()
    r1.title("fscore Graph")
    r1.geometry("500x500")
    f = Figure(figsize=(10,10),dpi=100)
    a = f.add_subplot(111)
    height =
[knn_fmeasure,nb_fmeasure,tree_fmeasure,svm_fmeasure,logistic_fmeasure,ensemble_fmea
sure,optimize_fmeasure]
    bars = ('KNN FScore', 'NB FScore','DT FScore','SVM FScore','LR FScore','RF
FScore','Optimize Ensemble FScore')
    y_pos = np.arange(len(bars))
    pps = a.bar(bars, height)
    #a.xticks(y_pos, bars)
    for p in pps:
        height = p.get_height()
        a.annotate('{:.4f}'.format(height),
        xy=(p.get_x() + p.get_width() / 2, height),
        xytext=(0, 3), # 3 points vertical offset
        textcoords="offset points",
        ha='center', va='bottom')
    canvas = FigureCanvasTkAgg(f,master=r1)
   # canvas.show()
    canvas.get_tk_widget().pack(side = BOTTOM,fill = BOTH,expand = True)
    #plt.show()
    r1.mainloop()
```

Accuracy Graph

```
def accuracyGraph():
    r1 = tkinter.Tk()
    r1.title("Accuracy Graph")
    r1.geometry("500x500")
    f = Figure(figsize=(10,10),dpi=100)
    a = f.add_subplot(111)
    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','RF ACC','Optimize
Ensemble ACC')
    y_pos = np.arange(len(bars))
    pps = a.bar(bars, height)
    #a.xticks(y_pos, bars)
    for p in pps:
        height = p.get_height()
        a.annotate('{:.4f}'.format(height),
        xy=(p.get_x() + p.get_width() / 2, height),
        xytext=(0, 3), # 3 points vertical offset
        textcoords="offset points",
        ha='center', va='bottom')
    canvas = FigureCanvasTkAgg(f,master=r1)
   # canvas.show()
    canvas.get_tk_widget().pack(side = BOTTOM,fill = BOTH,expand = True)
    #plt.show()
    r1.mainloop()
```

Causes Graph

```
def causesGraph():
    r1 = tkinter.Tk()
    r1.title("Causes Graph")
    r1.geometry("500x500")
    f = Figure(figsize=(10,10),dpi=100)
    ax1 = f.add_subplot(111)
    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)
    explode = (0.1, 0.1, 0.1, 0.1, 0.1 ,0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)
    ax1.pie(values, explode=explode, labels=labels, autopct='%1.2f%%',shadow=False,
startangle=155)
    ax1.axis('equal')
    canvas = FigureCanvasTkAgg(f,master=r1)
   # canvas.show()
    canvas.get_tk_widget().pack(side = BOTTOM,fill = BOTH,expand = True)
    #plt.show()
    r1.mainloop()
```

Dangerous Objects Extraction

```
#Dangerous objects extraction
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")
```

Tkinter module

```
#Tkinter window buttons
font = ('times', 16, 'bold')
title = Label(main, text='Analysis and Prediction of Occupational Accidents')
title.config(bg='dark goldenrod', fg='white')
title.config(font=font)
title.config(height=2, width=120)
title.place(x=0,y=5)

font1 = ('times', 10, 'bold')
upload = Button(main, text="Upload OSHA Dataset", command=upload)
upload.place(x=700,y=100)
upload.config(font=font1)

pathlabel = Label(main)
pathlabel.config(bg='DarkOrange1', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=700,y=140)

svmButton = Button(main, text="Run SVM Algorithm", command=SVM)
svmButton.place(x=700,y=180)
svmButton.config(font=font1)

knnButton = Button(main, text="Run KNN Algorithm", command=KNN)
knnButton.place(x=700,y=220)
knnButton.config(font=font1)

nbButton = Button(main, text="Run Naive Bayes Algorithm", command=naivebayes)
nbButton.place(x=700,y=260)
nbButton.config(font=font1)

treeButton = Button(main, text="Run Decision Tree Algorithm", command=decisionTree)
treeButton.place(x=700,y=300)
treeButton.config(font=font1)

lrButton = Button(main, text="Run Logistic Regression Algorithm",
command=logisticRegression)
lrButton.place(x=700,y=340)
lrButton.config(font=font1)


ensembleButton = Button(main, text="Run Random Forest Algorithm", command=ensemble)
ensembleButton.place(x=700,y=380)
ensembleButton.config(font=font1)

cnnButton = Button(main, text="Run Optimized Ensemble Algorithm",
command=optimizedEnsemble)
cnnButton.place(x=700,y=420)
cnnButton.config(font=font1)


graphButton = Button(main, text="Precision Graph", command=precisionGraph)
graphButton.place(x=700,y=460)
graphButton.config(font=font1)
recallButton = Button(main, text="Recall Graph", command=recallGraph)
recallButton.place(x=900,y=460)
recallButton.config(font=font1)
scoreButton = Button(main, text="Fscore Graph", command=fscoreGraph)
scoreButton.place(x=1060,y=460)
scoreButton.config(font=font1)
accButton = Button(main, text="Accuracy Graph", command=accuracyGraph)
accButton.place(x=700,y=500)
accButton.config(font=font1)
causesButton = Button(main, text="Causes Accidents Graph", command=causesGraph)
causesButton.place(x=900,y=500)
causesButton.config(font=font1)
predictButton = Button(main, text="Predict Accidents", command=predict)
predictButton.place(x=700,y=540)
predictButton.config(font=font1)

extractButton = Button(main, text="Chunker Dangerous Object Extraction",
command=extraction)
extractButton.place(x=900,y=540)
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='turquoise')
main.mainloop()
```

4.2 OUTPUT SCREENSHOTS

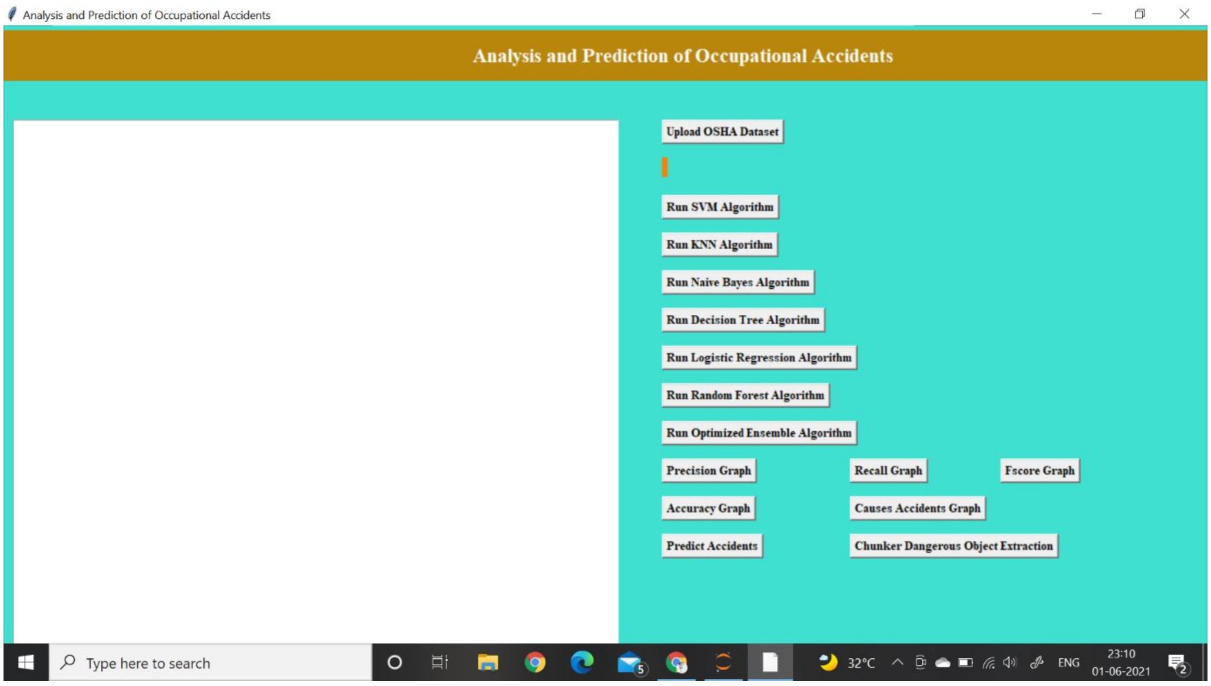
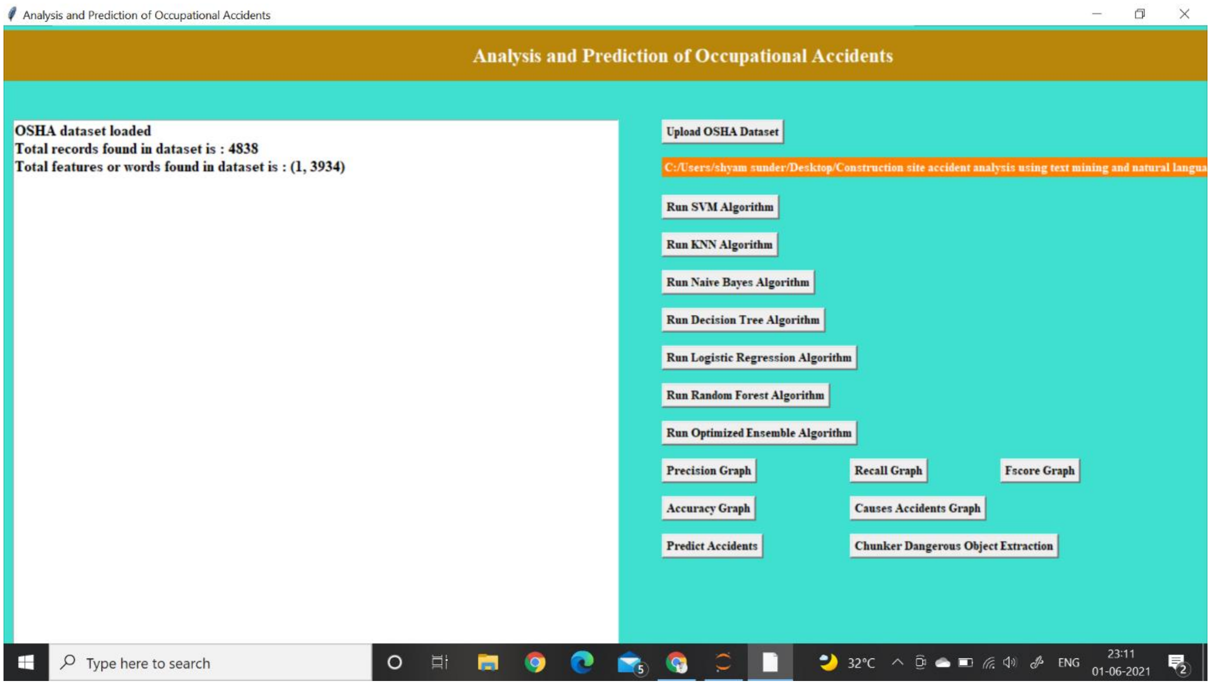

Analysis and Prediction of Occupational Accidents
Analysis and Prediction of Occupational Accidents
Upload OSHA Dataset
Run SVM Algorithm
Run KNN Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Run Logistic Regression Algorithm
Run Random Forest Algorithm
Run Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction
Type here to search
32°C
ENG
23:10
01-06-2021

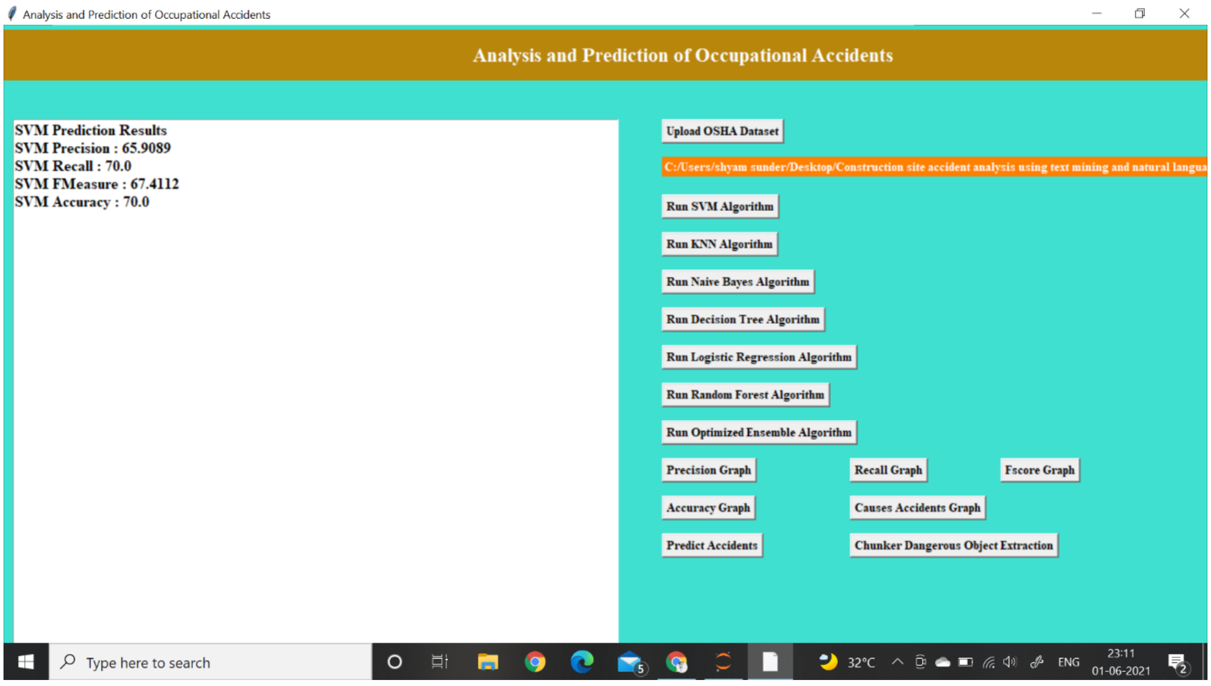
Fig 24. Main window


Analysis and Prediction of Occupational Accidents
Analysis and Prediction of Occupational Accidents
Upload OSHA Dataset
Run SVM Algorithm
Run KNN Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Run Logistic Regression Algorithm
Run Random Forest Algorithm
Run Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction
Type here to search
32°C
ENG
23:11
01-06-2021

```
OSHA dataset loaded
Total records found in dataset is : 4838
Total features or words found in dataset is : (1, 3934)
```

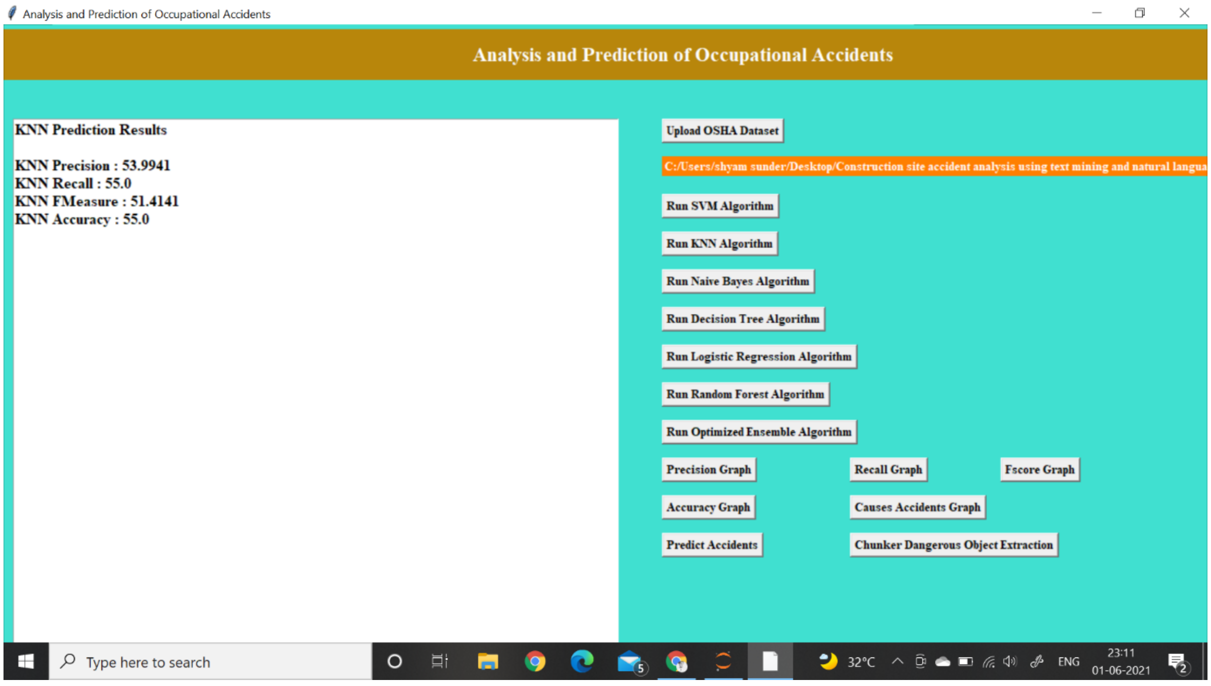
C:/Users/shyam sunder/Desktop/Construction site accident analysis using text mining and natural langua

Fig 25. Loading of data set


Analysis and Prediction of Occupational Accidents
Upload OSHA Dataset
C:/Users/shyam sunder/Desktop/Construction site accident analysis using text mining and natural langua
Run SVM Algorithm
Run KNN Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Run Logistic Regression Algorithm
Run Random Forest Algorithm
Run Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction

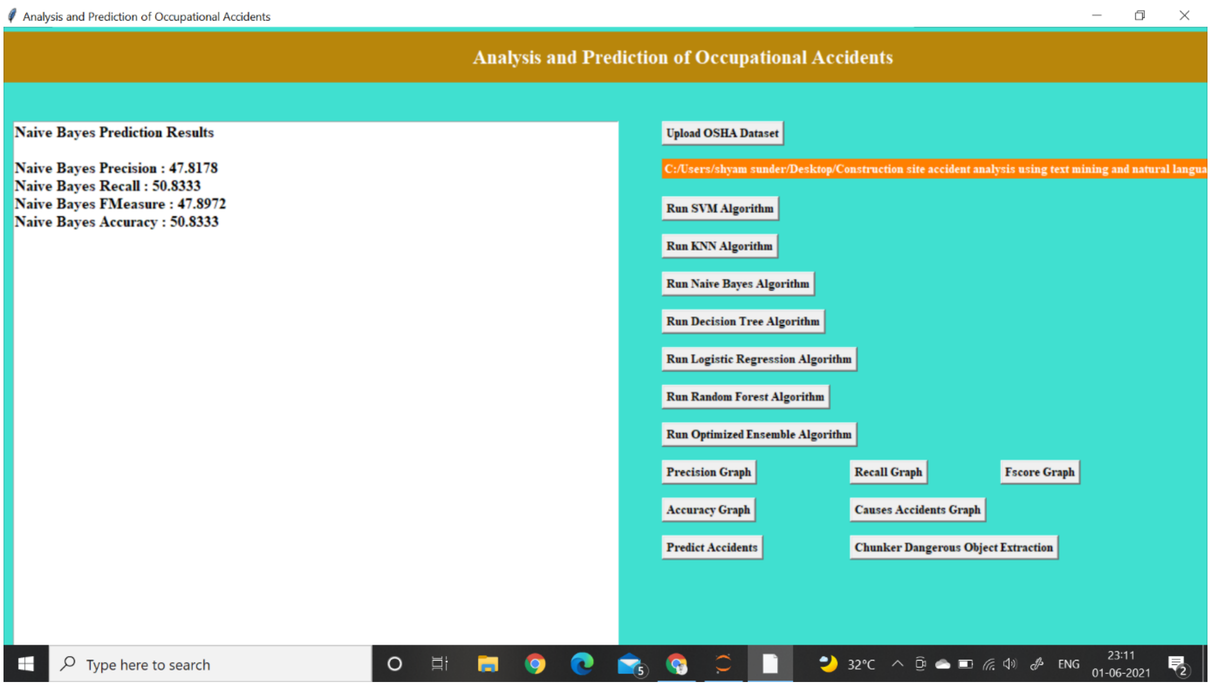
```
SVM Prediction Results
SVM Precision : 65.9089
SVM Recall : 70.0
SVM FMeasure : 67.4112
SVM Accuracy : 70.0
```

Fig 26. Result for SVM algorithm


Analysis and Prediction of Occupational Accidents
Upload OSHA Dataset
C:/Users/shyam sunder/Desktop/Construction site accident analysis using text mining and natural langua
Run SVM Algorithm
Run KNN Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Run Logistic Regression Algorithm
Run Random Forest Algorithm
Run Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction

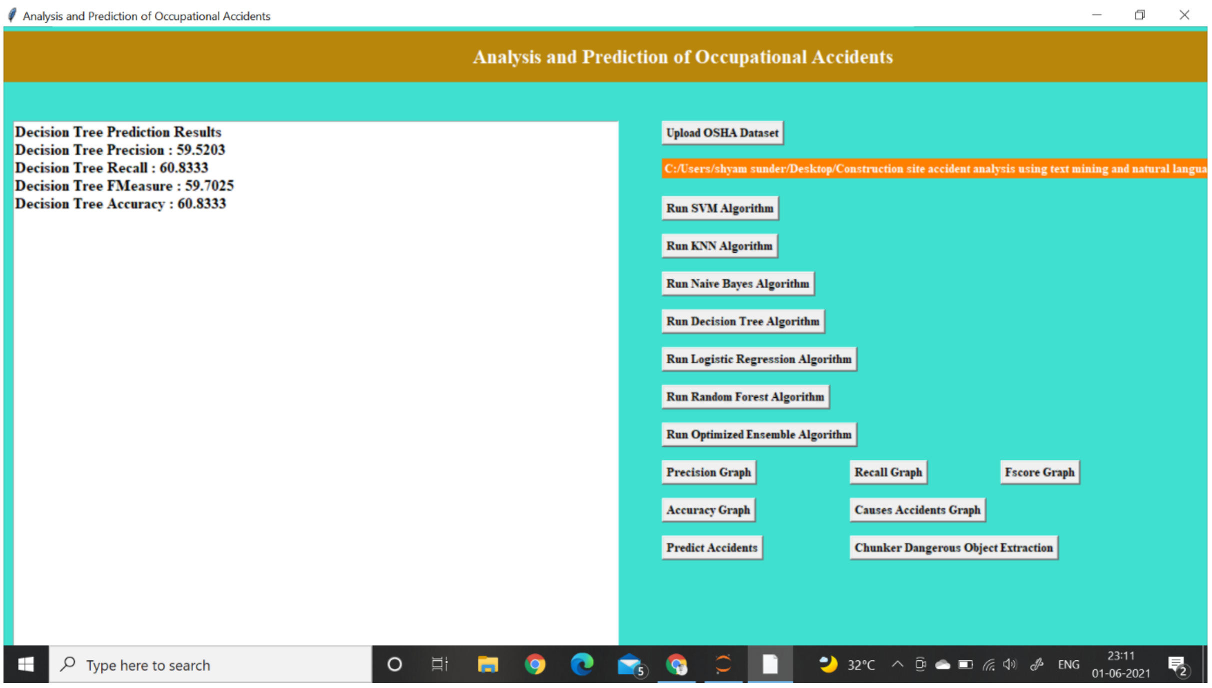
```
KNN Prediction Results

KNN Precision : 53.9941
KNN Recall : 55.0
KNN FMeasure : 51.4141
KNN Accuracy : 55.0
```

Fig 27. Result for KNN algorithm


Analysis and Prediction of Occupational Accidents
Upload OSHA Dataset
C:/Users/shyam sunder/Desktop/Construction site accident analysis using text mining and natural langua
Run SVM Algorithm
Run KNN Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Run Logistic Regression Algorithm
Run Random Forest Algorithm
Run Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction

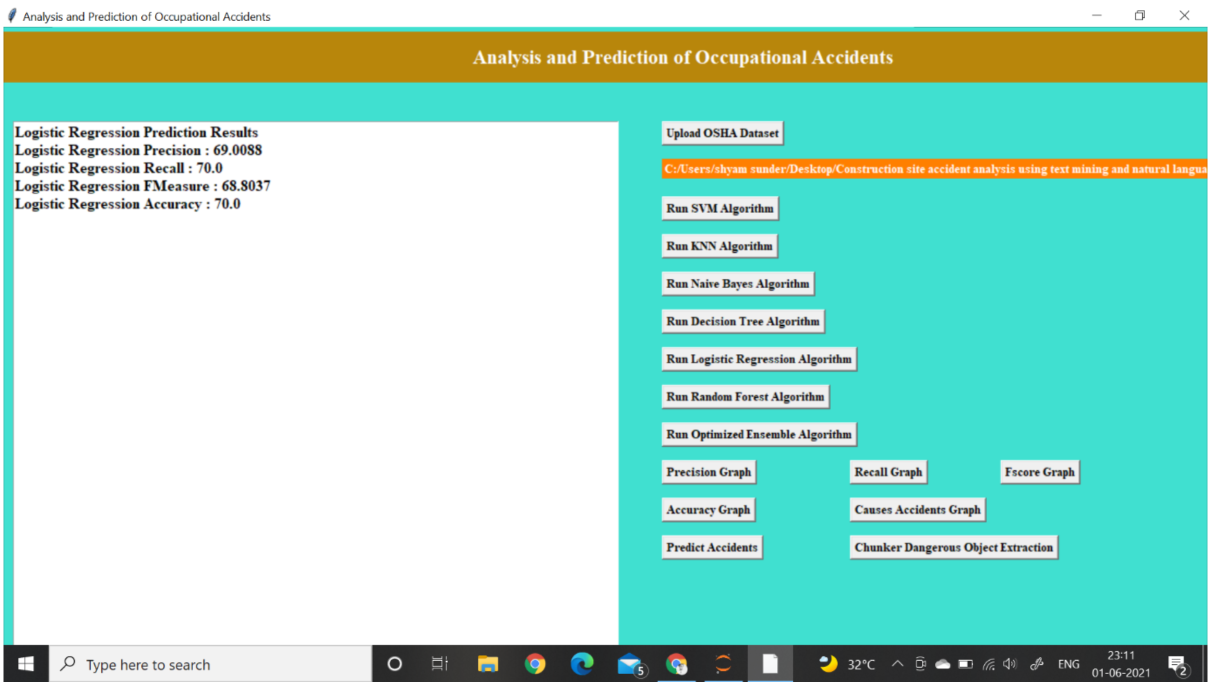
```
Naive Bayes Prediction Results

Naive Bayes Precision : 47.8178
Naive Bayes Recall : 50.8333
Naive Bayes FMeasure : 47.8972
Naive Bayes Accuracy : 50.8333
```

Fig 28. Results for naïve bayes algorithm.


Analysis and Prediction of Occupational Accidents
Upload OSHA Dataset
C:/Users/shyam sunder/Desktop/Construction site accident analysis using text mining and natural langua
Run SVM Algorithm
Run KNN Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Run Logistic Regression Algorithm
Run Random Forest Algorithm
Run Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction

```
Decision Tree Prediction Results
Decision Tree Precision : 59.5203
Decision Tree Recall : 60.8333
Decision Tree FMeasure : 59.7025
Decision Tree Accuracy : 60.8333
```

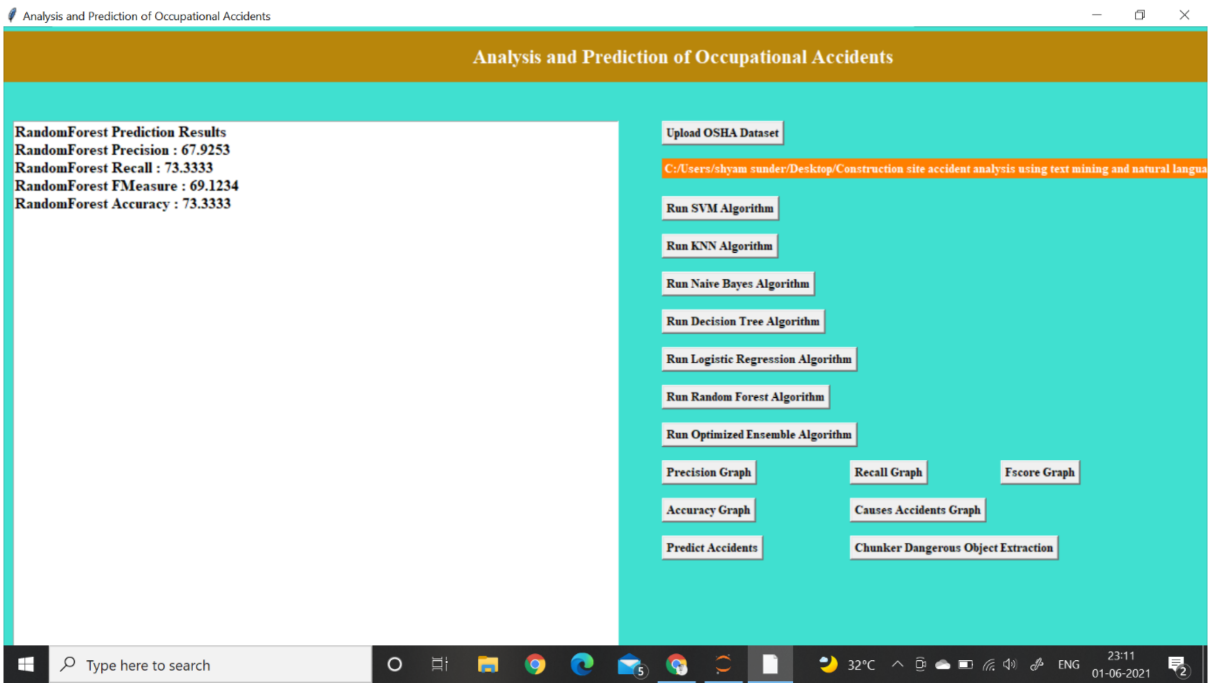
Fig 29. Decision tree algorithm



```
Logistic Regression Prediction Results
Logistic Regression Precision : 69.0088
Logistic Regression Recall : 70.0
Logistic Regression FMeasure : 68.8037
Logistic Regression Accuracy : 70.0
```

Analysis and Prediction of Occupational Accidents
Upload OSHA Dataset
C:/Users/shyam sunder/Desktop/Construction site accident analysis using text mining and natural langua
Run SVM Algorithm
Run KNN Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Run Logistic Regression Algorithm
Run Random Forest Algorithm
Run Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction

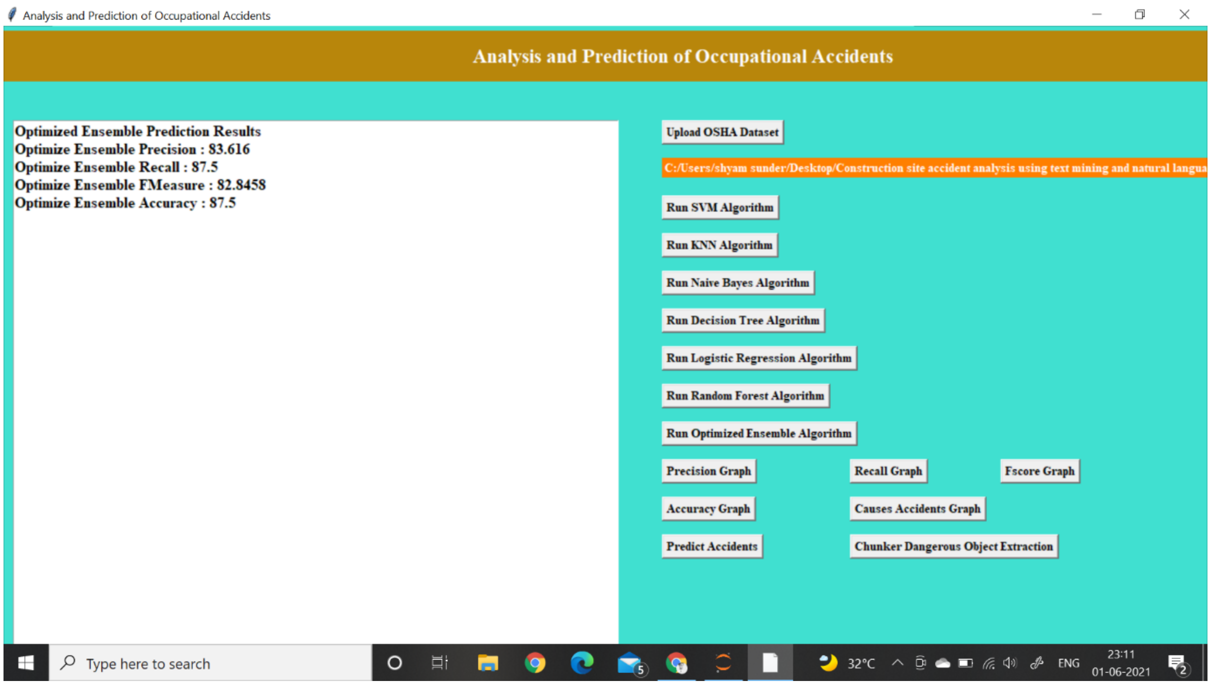
Fig 30. Logistic regression algorithm



```
RandomForest Prediction Results
RandomForest Precision : 67.9253
RandomForest Recall : 73.3333
RandomForest FMeasure : 69.1234
RandomForest Accuracy : 73.3333
```

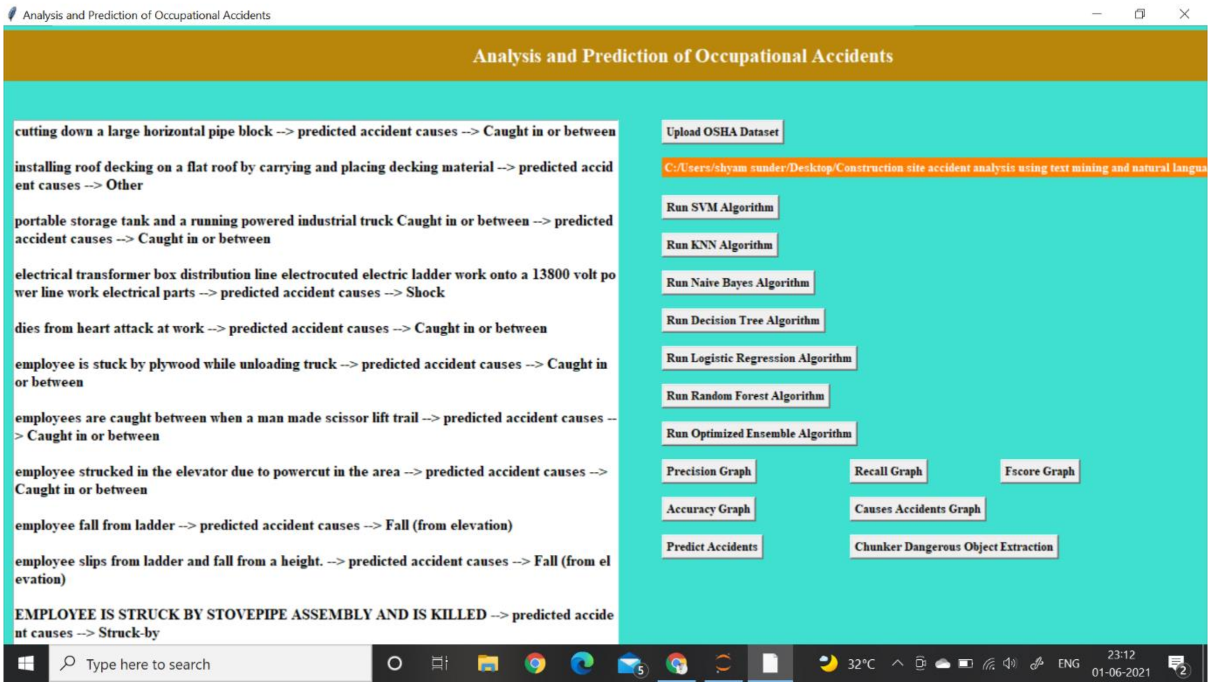
Analysis and Prediction of Occupational Accidents
Upload OSHA Dataset
C:/Users/shyam sunder/Desktop/Construction site accident analysis using text mining and natural langua
Run SVM Algorithm
Run KNN Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Run Logistic Regression Algorithm
Run Random Forest Algorithm
Run Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction

Fig 31. Ensemble algorithm


Analysis and Prediction of Occupational Accidents
Analysis and Prediction of Occupational Accidents
Upload OSHA Dataset
C:/Users/shyam sunder/Desktop/Construction site accident analysis using text mining and natural langua
Run SVM Algorithm
Run KNN Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Run Logistic Regression Algorithm
Run Random Forest Algorithm
Run Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction
Type here to search
32°C
ENG
23:11
01-06-2021

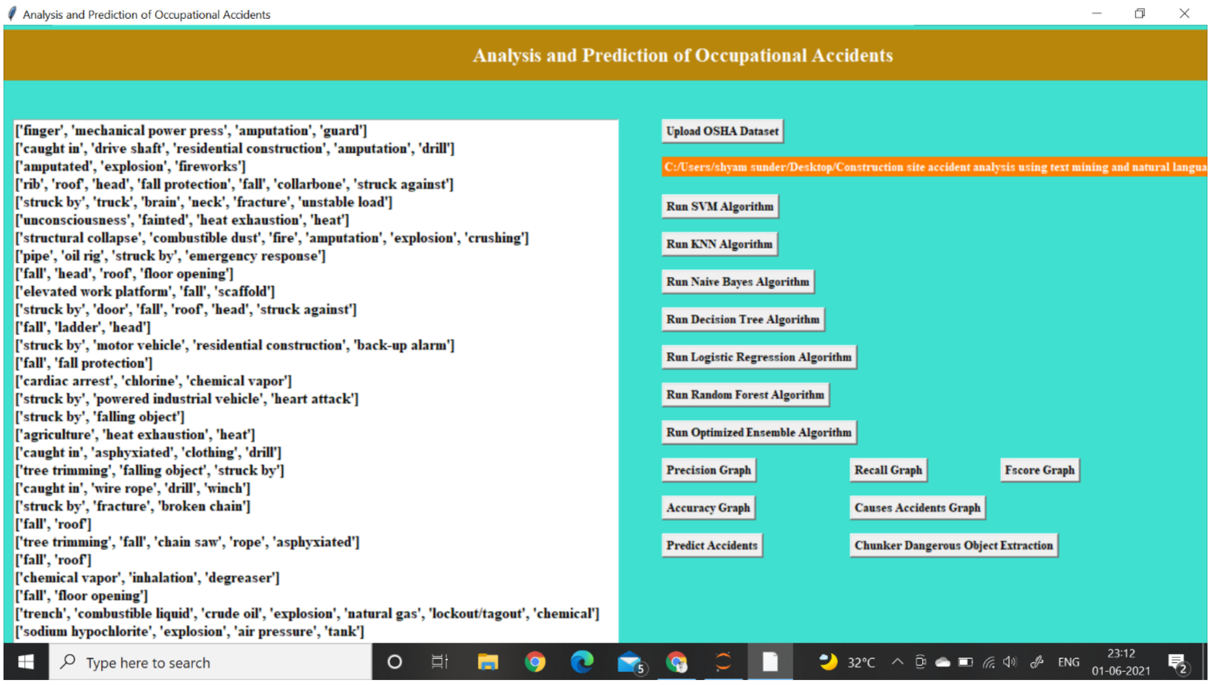
```
Optimized Ensemble Prediction Results
Optimize Ensemble Precision : 83.616
Optimize Ensemble Recall : 87.5
Optimize Ensemble FMeasure : 82.8458
Optimize Ensemble Accuracy : 87.5
```

Fig 32. Optimised ensemle prediction


Analysis and Prediction of Occupational Accidents
Analysis and Prediction of Occupational Accidents
Upload OSHA Dataset
C:/Users/shyam sunder/Desktop/Construction site accident analysis using text mining and natural langua
Run SVM Algorithm
Run KNN Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Run Logistic Regression Algorithm
Run Random Forest Algorithm
Run Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction
Type here to search
32°C
ENG
23:12
01-06-2021

```
cutting down a large horizontal pipe block --> predicted accident causes --> Caught in or between

installing roof decking on a flat roof by carrying and placing decking material --> predicted accid
ent causes --> Other

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

electrical transformer box distribution line electrocuted electric ladder work onto a 13800 volt po
wer line work electrical parts --> predicted accident causes --> Shock

dies from heart attack at work --> predicted accident causes --> Caught in or between

employee is stuck by plywood while unloading truck --> predicted accident causes --> Caught in
or between

employees are caught between when a man made scissor lift trail --> predicted accident causes --
> Caught in or between

employee strucked in the elevator due to powercut in the area --> predicted accident causes -->
Caught in or between

employee fall from ladder --> predicted accident causes --> Fall (from elevation)

employee slips from ladder and fall from a height. --> predicted accident causes --> Fall (from el
evation)

EMPLOYEE IS STRUCK BY STOVEPIPE ASSEMBLY AND IS KILLED --> predicted accide
nt causes --> Struck-by
```

Fig 33. Prediction of accidents.


Analysis and Prediction of Occupational Accidents
Analysis and Prediction of Occupational Accidents
Upload OSHA Dataset
C:/Users/shyam sunder/Desktop/Construction site accident analysis using text mining and natural langua
Run SVM Algorithm
Run KNN Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Run Logistic Regression Algorithm
Run Random Forest Algorithm
Run Optimized Ensemble Algorithm
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Causes Accidents Graph
Predict Accidents
Chunker Dangerous Object Extraction
Type here to search
32°C
ENG
23:12
01-06-2021

```
['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']
```

Fig 34. Dangerous object extraction.

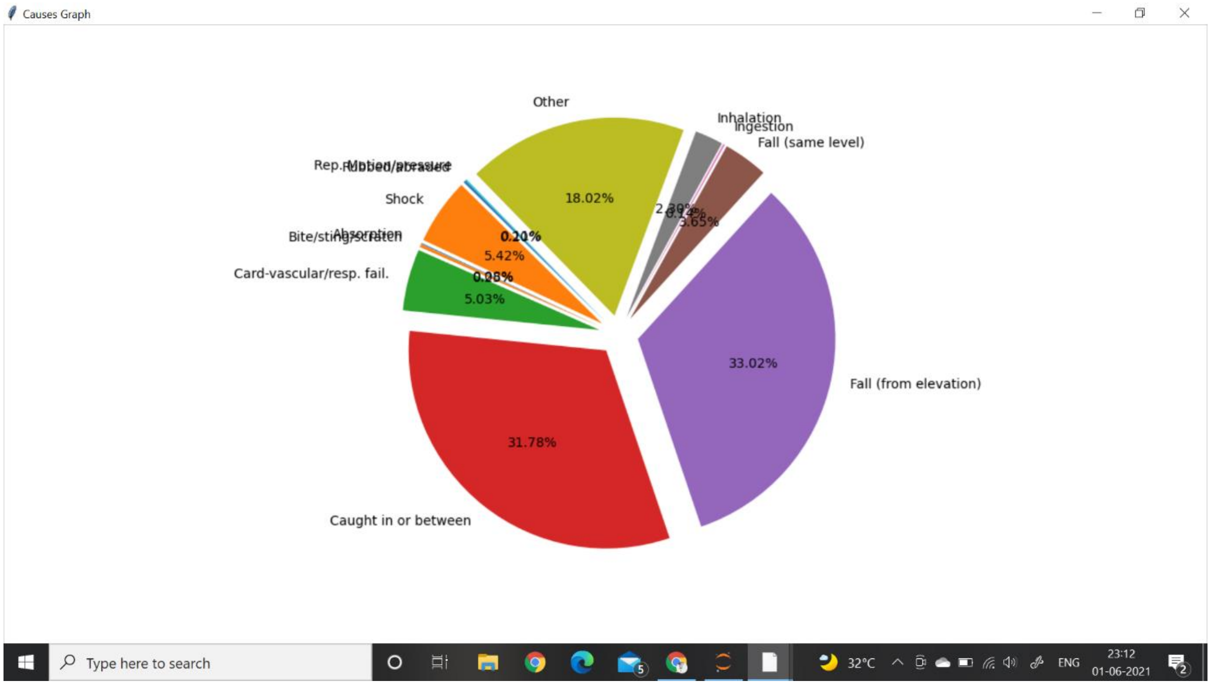


Fig 35. Cause of accidents.

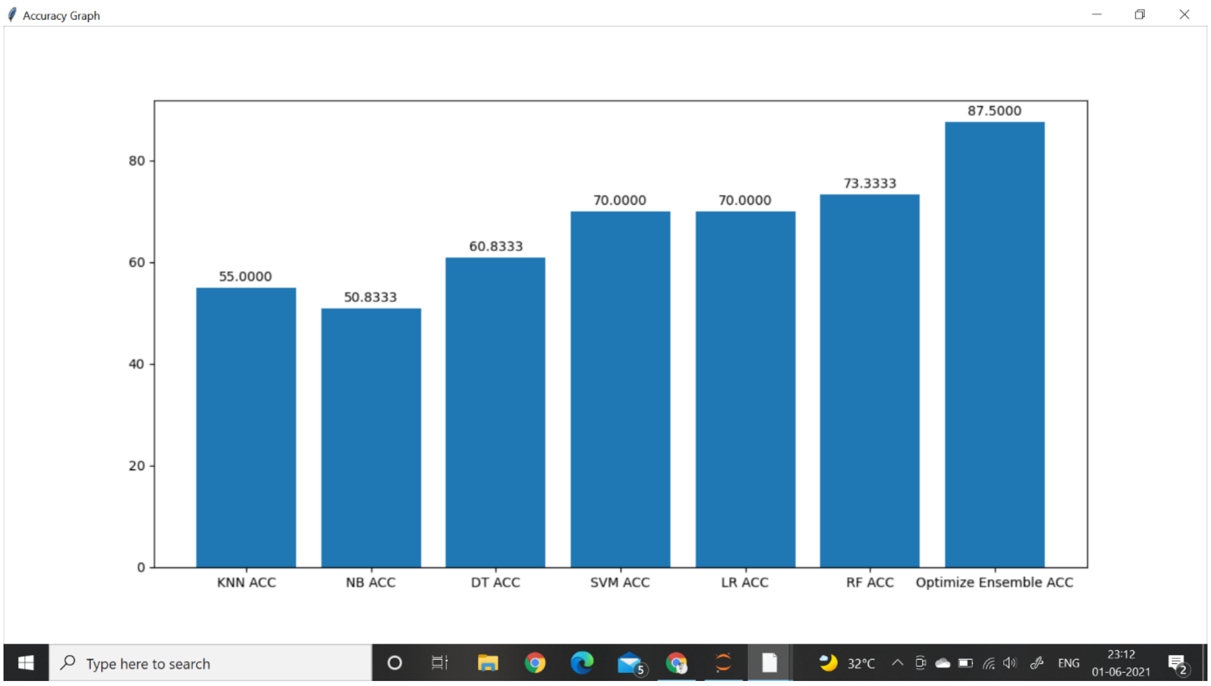


Fig 36. Accuracy graph

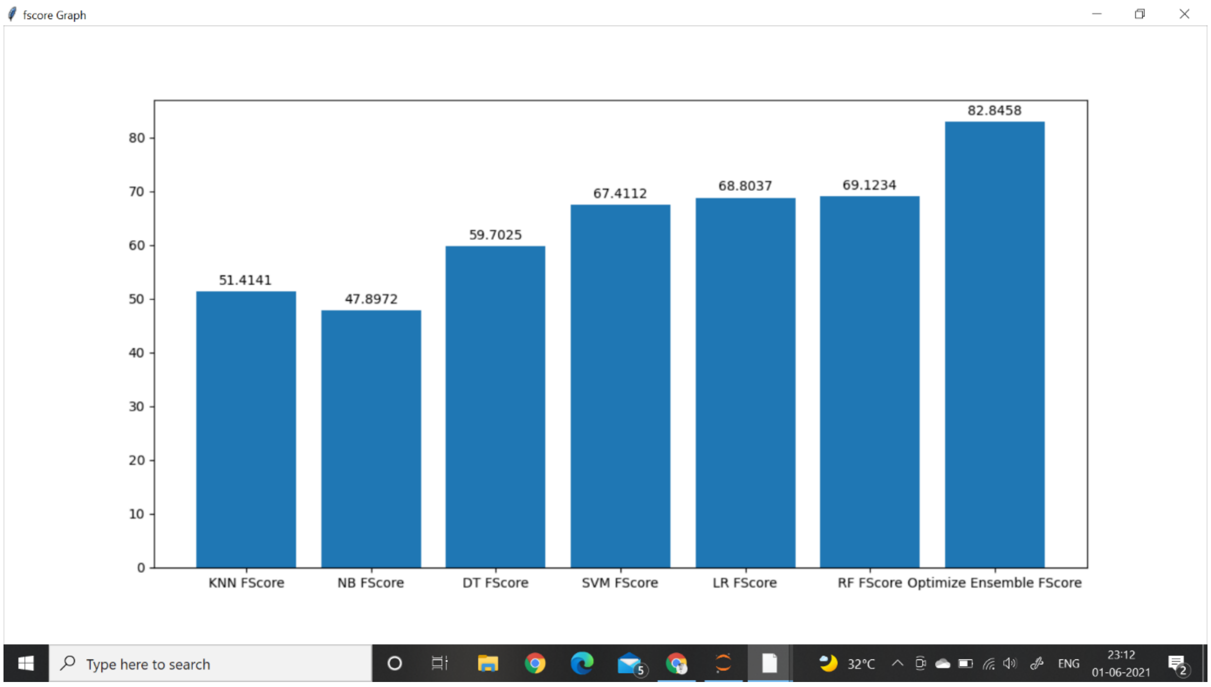


Fig 37. F-score graph

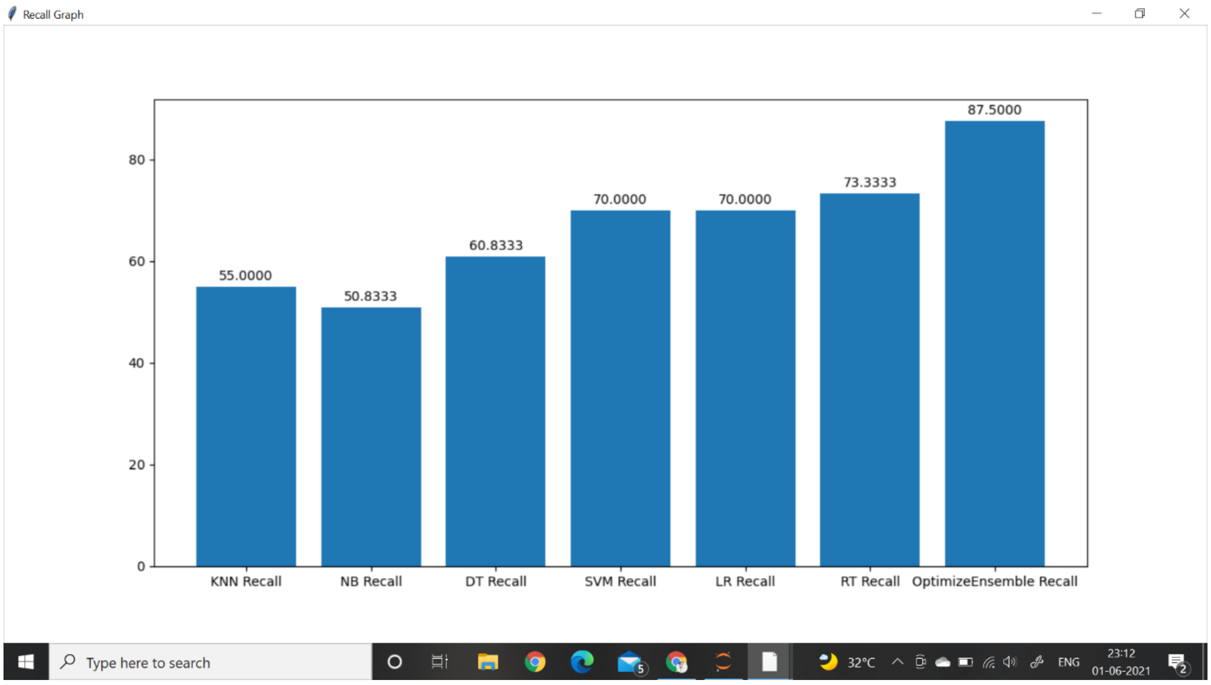


Fig 38. Recall graph.

5 TESTING

Testing is a set of activities that can be planned in advanced and conducted systematically. A strategy for software testing must accommodation low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements. Software testing is one element of verification and validation. Verification refers to the set of activities that ensure that software correctly implements as specific function. Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements. The main objective of software is testing to uncover errors. To fulfill this objective, a series of test steps unit, integration, validation and system tests are planned and executed. Each test step is accomplished through a series of systematic test technique that assist in the design of test cases. With each testing step, the level of abstraction with which software is considered is broadened.

Unit Testing

This testing method considers a module as single unit and checks the unit at interfaces and communicates with other modules rather than getting into details at statement level.

System testing

Here all the pre tested individual modules will be assembled to create the larger system and tests are carried out at system level to make sure that all modules are working in synchronous with each other.

Integration Testing

Testing is a major quality control measure employed during software development. Its basic function is to detect errors. Sub functions when combined may not produce than it is desired. Global data structures can represent the problems. Integrated testing is a systematic technique for constructing the program structure while conducting the tests.

Regression testing

Each time a new module is added as a part of integration as the software changes. Regression testing is an actually that helps to ensure changes that do not introduce unintended behavior as additional errors.

6. CONCLUSION

Analyzing the construction accident reports leads to valuable knowledge of what went wrong in the past in order to prevent future accidents. To be more specific, accident causes classification is essential as prevention strategies should be developed based on different causes accordingly. Besides, identification of dangerous objects plays a crucial role in improving the safety of the working environment as well, as preventive actions can be implemented to eliminate or mitigate the potential risks of identified objects. However, manual classification of accident reports and investigation of dangerous objects involved in accidents are time consuming and labor intensive. In this work, an ensemble model with optimized weights is proposed for construction accident causes classification. The results show that the proposed model outperforms other single model in terms of the average weighted F1 score. Further, the proposed model is proved to be more robust to the cases of low support. Moreover, a rule based chunker approach is explored to identify the common objects which cause the accidents.

Therefore, the aforementioned labor intensive tasks are effectively automated by the proposed approaches. Besides, the proposed approaches support the informed culture and play an important role in improving the safety information system proposed by Reason which enhance the construction site safety in the long run.

Several possible future improvements can be considered, for example, data balancing techniques such as under sampling, oversampling or a combination of both can be applied. Compiling a stop words list specific to construction accident domain which reduces stop
words more accurately is also an approach can be considered to improve the data quality.

7. REFERENCES.

- https://doi.org/10.1061/(ASCE)CF.1943-5509.
- https://doi.org/10.1016/j.apergo.2004.12.002
- https://doi.org/10.1016/j.aap.2013.09.012
- https://doi.org/10.1016/S0001-4575(02)00146-X