A Project Report
on

# Accident Severity Prediction using Machine Learning Algorithms

submitted in partial fulfillment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

| | |
|---|---|
| 16WH1A0599 | Ms. S.Vaishnavi |
| 16WH1A05A8 | Ms. V.Yashaswini Priyanka |
| 16WH1A05B6 | Ms. T R Manisha Reddy |

under the esteemed guidance of

Ms.G.Shanti
Assistant Professor

Department of Computer Science and Engineering
BVRIT HYDERABAD
College of Engineering for Women
(NAAC Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

April, 2020

# DECLARATION

We hereby declare that the work presented in this project entitled **“ACCIDENT SEVERITY PREDICTION USING MACHINE LEARNING ALGORITHMS”** 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 Ms. G.Shanti, Assistant Professor, Department of CSE.

Sign. with date:
**Ms. S.Vaishnavi**
**(16WH1A0599)**

Sign. with date:
**Ms. V.Yashaswini Priyanka**
**(16WH1A05A8)**

Sign. with date:
**Ms. T R Manisha Reddy**
**(16WH1A05B6)**

**BVRIT HYDERABAD**
**College of Engineering for Women**
**(NAAC Accredited – EEE, ECE, CSE and IT)**
**(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)**
**Bachupally, Hyderabad – 500090**

**Department of Computer Science and Engineering**

## Certificate

This is to certify that the Project Work report on "**ACCIDENT SEVERITY PREDICTION USING MACHINE LERANING ALGORITHMS**" is a bonafide work carried out by Ms. S.Vaishnavi (16WH1A0599) ; Ms. V.Yashaswini Priyanka (16WH1A05A8) ; Ms.T R Manisha Reddy(16WH1A05B6) 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.

| | |
|---|---|
| **Head of the Department** | **Guide** |
| **Dr. Ch. Srinivasulu** | **Ms. G.Shanti** |
| **Professor and HoD,** | **Assistant Professor** |
| **Department of CSE** | |

**External Examiner**

## Acknowledgements

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

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

We are extremely thankful and indebted to our internal guide, **Ms. G.Shanti, Assistant Professor**, Department of CSE**, BVRIT HYDERABAD College of Engineering for Women** for his constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project Coordinator, all the faculty and staff of **CSE** Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

**Ms. S.Vaishnavi**

**(16WH1A0599)**

**Ms. V.Yashaswini Priyanka**

**(16WH1A05A8)**

**Ms. T R Manisha Reddy**

**(16WH1A05B6)**

# Contents

## ABSTRACT

Road Accidents are a serious concern for the majority of nations around the world because accidents can cause severe injuries and fatalities. In recent years, there is an increase in the research's attention to determine the significantly affect the severity of the injuries which is caused due to the road accidents. Accurate and comprehensive accident records are the basis of accident analysis. The effective use of accident records depends on some factors, like the accuracy of the data, record retention, and data analysis. There are many approaches applied to this scenario to study this problem. A recent study illustrated that the residential and shopping sites are more hazardous than village areas as might have been predicted, the frequencies of the casualties were higher near the zones of residence possibly because of the higher exposure. A study revealed that the casualty rates among the residential areas are classified as relatively deprived and significantly higher than those from relatively affluent areas.

According to the World Health Organization's Global Status Report, approximately 1.25 million people deaths happen per year are because of road accident injuries, and most fatality rates were in lower income countries. Our motivation is to predict the accident severity of Telangana roads, which will play a crucial factor for traffic control authorities to take proactive precautionary measures.

# LIST OF FIGURES

# 1. INTRODUCTION

There is a huge impact on the society due to traffic accidents where there is a great cost of fatalities and injuries. In recent years, there is an increase in the research's attention to determine the significantly affect the severity of the injuries which is caused due to the road accidents. Accurate and comprehensive accident records are the basis of accident analysis. The effective use of accident records depends on some factors, like the accuracy of the data, record retention, and data analysis. There are many approaches applied to this scenario to study this problem. A recent study illustrated that the residential and shopping sites are more hazardous than village areas as might have been predicted, the frequencies of the casualties were higher near the zones of residence possibly because of the higher exposure. A study revealed that the casualty rates among the residential areas are classified as relatively deprived and significantly higher than those from relatively affluent areas.

According to the World Health Organization's Global Status Report, approximately 1.25 million people deaths happen per year are because of road accident injuries, and most fatality rates were in lower income countries. Our motivation is to predict the accident severity of telangana roads, which will play a crucial factor for traffic control authorities to take proactive precautionary measures. In addition, the dataset we chose was rarely solved from a prediction point of view, so we took this opportunity to predict the severity of the accident. Also, we got a highly imbalanced dataset.

## 1.1 Objective

The purpose of this project is to predict the severity of an accident by training an efficient machine learning model with the help of existing accidents data from telangana road records. This project is majorly focussed on predicting rarer classes accurately such as Serious and Fatal.

## 1.2 Organization of project

As the project is to predict rarer classes such as serious and Fatal.

We divided our project into 2 major modules.

- Firstly, cross validate the given dataset as it is highly imbalanced.
- Secondly, train the model for a given dataset, the model can be any one out of the seven machine learning algorithms which we decided to use.
- Finally, compare all the models and choose the best one.

# 2. Theoretical Analysis of proposed project

## 2.1 System Specifications and features

HARDWARE :

Operating System : Linux/Ubuntu

Processor : Intel core i7

CPU speed : 2.20 GHZ

Memory (RAM) : 8GB

SOFTWARE :

Programming Language : Python 3.6.7

Libraries: sklearn, pandas, ploty, numpy

Classes : Slight, Serious, Fatal

### 2.1.1 Technology Description

**Python**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages.

- **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 is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**Installation of Python**

- First and foremost step is to open a browser and open https://www.python.org/downloads/windows/.
- Underneath the Python Releases for Windows find Latest Python 3 Release – Python 3.7.4 (latest stable release as of now is Python 3.7.4).
- This page moves to Files and clicks on Windows x86-64 executable installer for 64-bit or Windows x86 executable installer for 32-bit.
- Run the Python Installer from downloads folder.Make sure to mark add Python 3.7 to PATH otherwise you will have to do it explicitly.
- It will start installing python on windows.After installation is complete click on Close.

**Sklearn**

Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machines, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like Numpy and Scipy. Learns features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

**Important features of scikit-learn:**

- Simple and efficient tools for data mining and data analysis. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, etc.

- Accessible to everybody and reusable in various contexts.
- Built on the top of NumPy, SciPy, and matplotlib.
- Open source, commercially usable – BSD license.

**Installation of Sklearn**

- Install the 64bit version of Python 3, for instance from https://www.python.org.
- Then run:pip install -U  scikit- learn

**Pandas**

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language.

Pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet.
- Ordered and unordered (not necessarily fixed-frequency) time series data.

- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels.
- Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure.

In order to use Pandas in your Python IDE (Integrated Development Environment) like Jupyter Notebook it should be imported.

1. **import pandas as pd**
2. **import numpy as np**

When you want to use Pandas for data analysis, you'll usually use it in one of three different ways:

- Convert a Python's list, dictionary or Numpy array to a Pandas data frame.
- Open a local file using Pandas, usually a CSV file, but could also be a delimited text file (like TSV), Excel, etc.
- Open a remote file or database like a CSV or a JSONon a website through a URL or read from a SQL table/database.

Different types of commands in pandas when a file is opened.

- **pd.read_filetype()**
- **pd.DataFrame()**

**Viewing and Inspecting Data**

- df.mean() Returns the mean of all columns
- df.corr() Returns the correlation between columns in a data frame
- df.count() Returns the number of non-null values in each data frame column

- · df.max()Returns the highest value in each column
- df.min()Returns the lowest value in each column
- df.median()Returns the median of each column
- df.std()Returns the standard deviation of each column

**Installation of Pandas**

- On command prompt type the command “pip install pandas.”
- Wait for the downloads to be over and once it is done you will be able to run Pandas inside your Python programs on Windows.

**Numpy**

Numpy is one such powerful library for array processing along with a large collection of high-level mathematical functions to operate on these arrays. These functions fall into categories like Linear Algebra, Trigonometry, Statistics, Matrix manipulation, etc.

Getting NumPy

NumPy’s main object is a homogeneous multidimensional array. Unlike python’s array class which only handles one-dimensional array, NumPy’s nd array class can handle multidimensional array and provides more functionality. NumPy’s dimensions are known as axes. For example, the array below has 2 dimensions or 2 axes namely rows and columns. Sometimes dimension is also known as a rank of that particular array or matrix.

Importing NumPy

NumPy is imported using the following command. Note here np is the convention followed for the alias so that we don't need to write numpy every time.

- **import numpy as np**

NumPy is the basic library for scientific computations in Python and this article illustrates some of its most frequently used functions. Understanding NumPy is the first major step in the journey of machine learning and deep learning.

**Installation of Numpy**

```
import sys
!conda install --yes --prefix {sys.prefix} numpy
```

**Plotly**

Plotly's Python graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts.

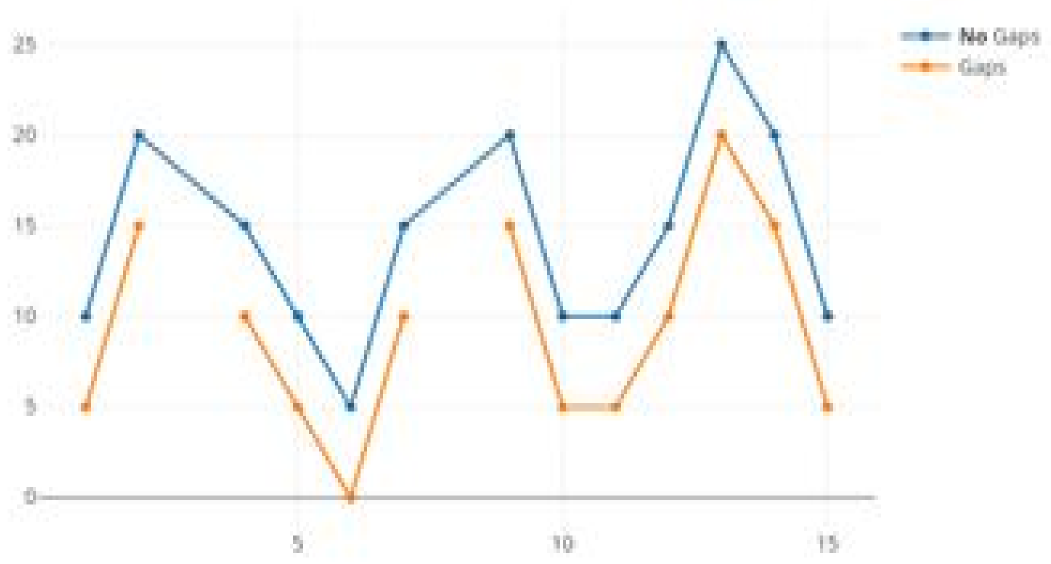
Plotly.py is free and open source.

Plotly's python API can be used to plot inside your Jupyter Notebook by calling plotly.plotly.iplot() or plotly.offline.iplot() if working offline. Plotting in the notebook gives you the advantage of keeping your data analysis and plots in one place. Now we can do a bit of interactive plotting. Head to the Plotly getting started page to learn how to set your credentials.

Calling the plot with iplot automatically generates an interactive version of the plot inside the Notebook in an iframe

**Installation of Ploty**

Install plotly.py 3.0.0 today with pip install plotly (or pip install plotly --upgrade) if plotly is already there in the past.



## 2.2 Architecture Diagram

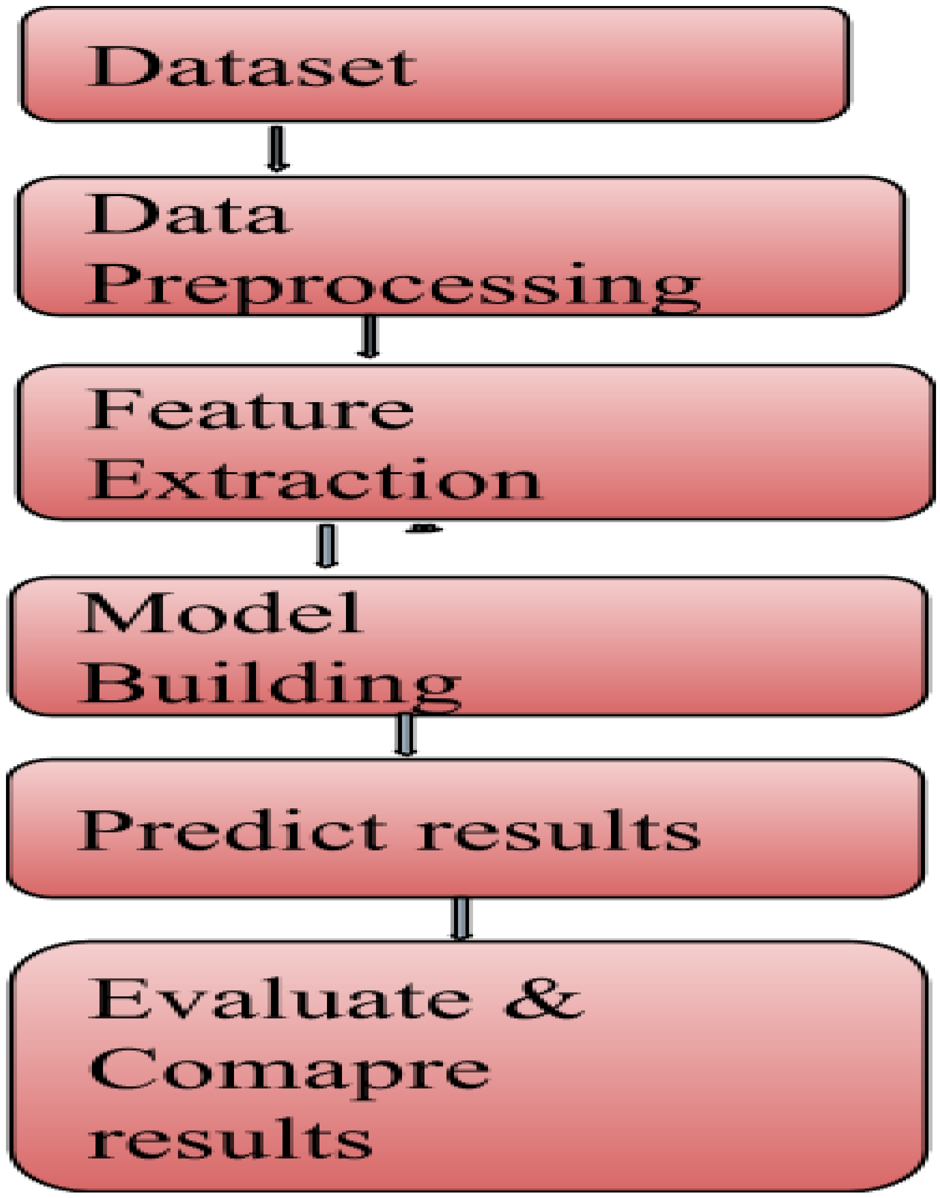


fig 2.2.1

## 2.3 Software Requirements Specifications

| Feature F-01 | |
|---|---|
| Priority | Essential – Telangana Road Accident Severity Features\| Expected-Classifying the input as serious,fatal or slight |
| Effort | Months – 1 \| Weeks – 0 \| Days – 0 \| Hours – 0 |
| User(s) | Users for this feature are common people,uses machine learning model to identify severity of accidents. |
| Description | • Input :– Accident Severity based features<br>• Desired output :– Indicates whether accident is serious,fatal or slight related to input given. |

Table 2.3.1

# 3. Basics for Supervised Machine Learning

## 3.1 MACHINE LEARNING

Machine Learning is a system that can learn from example through self-improvement and without being explicitly coded by programmers. The breakthrough comes with the idea that a machine can singularly learn from the data (i.e., example) to produce accurate results.

Machine learning combines data with statistical tools to predict an output. This output is then used by corporates to make actionable insights. Machine learning is closely related to data mining and Bayesian predictive modeling. The machine receives data as input, and uses an algorithm to formulate answers.

A typical machine learning task is to provide a recommendation. For those who have a Netflix account, all recommendations of movies or series are based on the user's historical data. Tech companies are using unsupervised learning to improve the user experience with personalizing recommendations.

Machine learning is also used for a variety of tasks like fraud detection, predictive maintenance, portfolio optimization, automatizes tasks and so on.

### 3.1.1 Machine Learning vs. Traditional Programming

Traditional programming differs significantly from machine learning. In traditional programming, a programmer codes all the rules in consultation with an expert in the industry for which software is being developed. Each rule is based on a logical foundation; the machine will execute an output following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.

### 3.1.2 How does Machine learning work?

Machine learning is the brain where all the learning takes place. The way the machine learns is similar to the human being. Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same. To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the outcome. However, like a human, if its feed a previously unseen example, the machine has difficulties to predict.

The core objective of machine learning is the learning and inference. First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the data. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a feature vector. You can think of a feature vector as a subset of data that is used to tackle a problem.

The machine uses some fancy algorithms to simplify the reality and transform this discovery into a model. Therefore, the learning stage is used to describe the data and summarize it into a model.

For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood to go to a fancy restaurant. It turns out the machine finds a positive relationship between wage and going to a high-end restaurant: This is the model Inferring

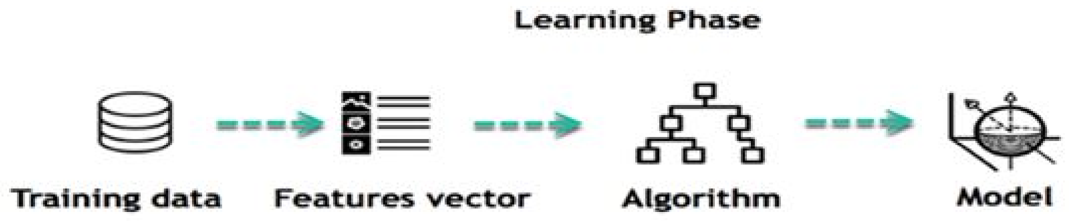
When the model is built, it is possible to test how powerful it is on never-seen-before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data

.

The life of Machine Learning programs is straightforward and can be summarized in the following points:

1. Define a question
2. Collect data
3. Visualize data
4. Train algorithm
5. Test the Algorithm
6. Collect feedback
7. Refine the algorithm
8. Loop 4-7 until the results are satisfying
9. Use the model to make a prediction

Once the algorithm gets good at drawing the right conclusions, it applies that knowledge to new sets of data.



### 3.1.3 Machine learning Algorithms and where they are used?

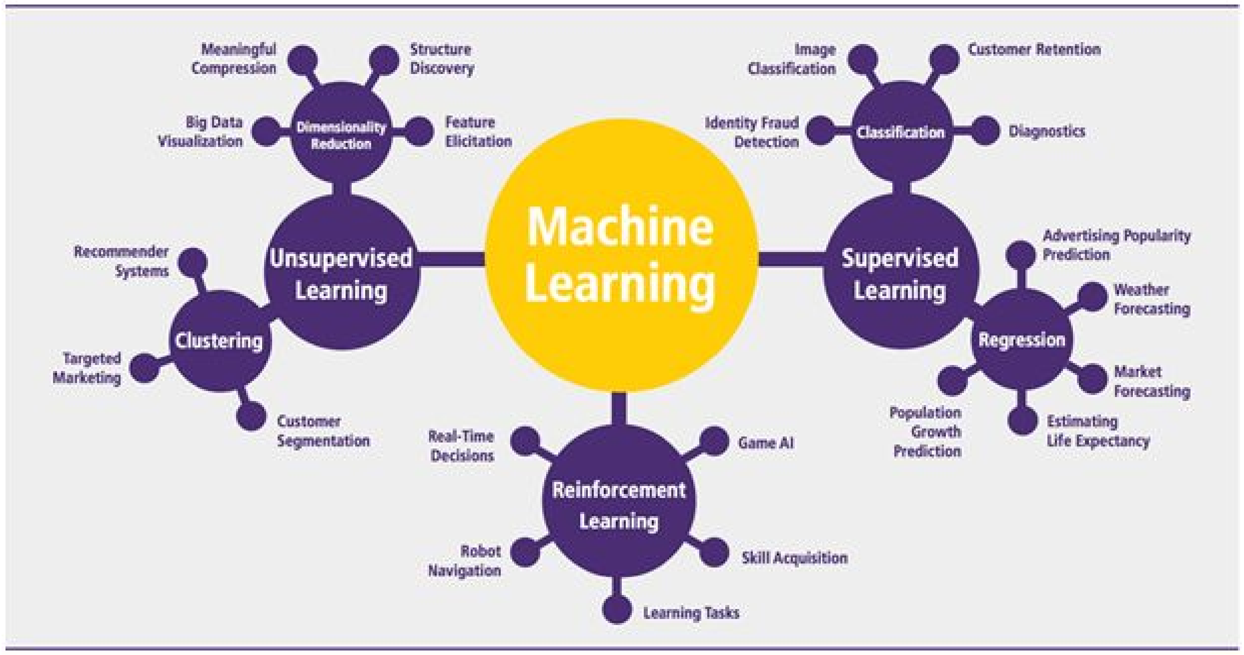


fig 3.1.3.1

Machine learning can be grouped into two broad learning tasks: Supervised and Unsupervised. There are many other algorithms

**Supervised learning**

An algorithm uses training data and feedback from humans to learn the relationship of given inputs to a given output. For instance, a practitioner can use marketing expense and weather forecast as input data to predict the sales of cans.

You can use supervised learning when the output data is known. The algorithm will predict new data.

There are two categories of supervised learning:

- Classification task
- Regression task

**Classification**

Imagine you want to predict the gender of a customer for a commercial. You will start gathering data on the height, weight, job, salary, purchasing basket, etc. from your customer database. You know the gender of each of your customer, it can only be male or female. The objective of the classifier will be to assign a probability of being a male or a female (i.e., the label) based on the information (i.e., features you have collected). When the model learned how to recognize male or female, you can use new data to make a prediction. For instance, you just got new information from an unknown customer, and you want to know if it is a male or female. If the classifier predicts male = 70%, it means the algorithm is sure at 70% that this customer is a male, and 30% it is a female.

The label can be of two or more classes. The above example has only two classes, but if a classifier needs to predict object, it has dozens of classes (e.g., glass, table, shoes, etc. each object represents a class)

**Regression**

When the output is a continuous value, the task is a regression. For instance, a financial analyst may need to forecast the value of a stock based on a range of features like equity, previous stock performances, macroeconomic index. The system will be trained to estimate the price of the stocks with the lowest possible error.

| **Algorithm Name** | **Description** | **Type** |
|---|---|---|
| Linear regression | Finds a way to correlate each feature to the output to help predict future values. | Regression |
| Logistic regression | Extension of linear regression that's used for classification tasks. The output variable 3is binary (e.g., only black or white) rather than continuous (e.g., an infinite list of potential colors) | Classification |

| | | |
|---|---|---|
| Decision tree | Highly interpretable classification or regression model that splits data-feature values into branches at decision nodes (e.g., if a feature is a color, each possible color becomes a new branch) until a final decision output is made | Regression Classification |
| Naive Bayes | The Bayesian method is a classification method that makes use of the Bayesian theorem. The theorem updates the prior knowledge of an event with the independent probability of each feature that can affect the event. | Regression Classification |
| Support vector machine | Support Vector Machine, or SVM, is typically used for the classification task. SVM algorithm finds a hyperplane that optimally divided the classes. It is best used with a non-linear solver. | Regression (not very common) Classification |
| Random forest | The algorithm is built upon a decision tree to improve the accuracy drastically. Random forest generates many times simple decision trees and uses the 'majority vote' method to decide on which label to return. For the classification task, the final prediction will be the one with the most vote; while for the regression task, the average prediction of all the trees is the final prediction. | Regression Classification |
| AdaBoost | Classification or regression technique that uses a multitude of models to come up with a decision but weighs them based on their accuracy in predicting the outcome | Regression Classification |
| Gradient-boosting trees | Gradient-boosting trees is a state-of-the-art classification/regression technique. It is focusing on the error committed by the previous trees and tries to correct it. | Regression Classification |

**Table 3.1.3.2**

## Application of Machine learning

### Augmentation:

- Machine learning, which assists humans with their day-to-day tasks, personally or commercially without having complete control of the output. Such machine learning is used in different ways such as Virtual Assistant, Data analysis, software solutions. The primary user is to reduce errors due to human bias.

### Automation:

- Machine learning, which works entirely autonomously in any field without the need for any human intervention. For example, robots performing the essential process steps in manufacturing plants.

### Finance Industry

- Machine learning is growing in popularity in the finance industry. Banks are mainly using ML to find patterns inside the data but also to prevent fraud**.**

### Government organization

- The government makes use of ML to manage public safety and utilities. Take the example of China with the massive face recognition.

### Healthcare industry

- Healthcare was one of the first industry to use machine learning with image detection.

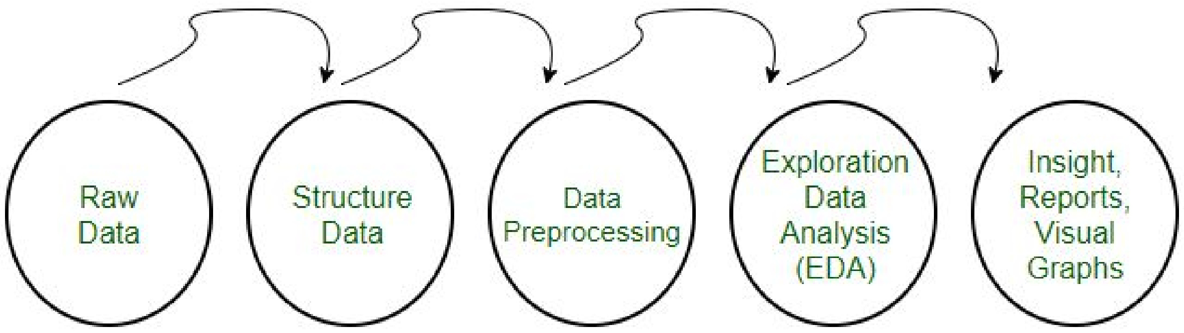
### Marketing

- Broad use of AI is done in marketing thanks to abundant access to data. Before the age of mass data, researchers develop advanced mathematical tools like Bayesian analysis to estimate the value of a customer. With the boom of data, marketing department relies on AI to optimize the customer relationship and marketing campaign**.**

## 3.2 Preprocessing the Dataset

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

fig 3.2.1


Raw Data
Structure Data
Data Preprocessing
Exploration Data Analysis (EDA)
Insight, Reports, Visual Graphs

**Need of Data Preprocessing:**

For achieving better results from the applied model in Machine Learning projects the format of the data has to be in a proper manner. Some specified Machine Learning models need information in a specified format, for example, Random Forest algorithm does not support null values, therefore to execute random forest algorithm null values have to be managed from the original raw data set.

Another aspect is that dataset should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in one data set, and best out of them is chosen.

## 3.3 Exploratory Data Analysis

Exploratory Data Analysis or (EDA) is understanding the data sets by summarizing their main characteristics and  gaining a better understanding of data aspects like:

– main features of data

– variables and relationships that hold between them

– identifying which variables are important for our problem

Exploratory Data Analysis is a crucial step before you jump to machine learning or modeling your data. By doing this you can get to know whether the selected features are good enough to model, are all the features required, are there any correlations based on which we can either go back to the Data Preprocessing step or move on to modeling

**Handling Missing Values**:

Data in the real world are rarely clean and homogeneous. Data can either be missing during data extraction or collection due to several reasons. Missing values need to be handled carefully because they reduce the quality of any of our performance metrics. It can also lead to wrong prediction or classification and can also cause a high bias for any given model being used.

The  most common method of handling missing values is a process whereby missing values are replaced with a test statistic like mean, median or mode of the particular feature the missing value belongs to.

## 3.4 Algorithms Overview

### 3.4.1 K-Nearest Neighbours

K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.

It is widely disposable in real-life scenarios since it is non-parametric, meaning it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data).

We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

,

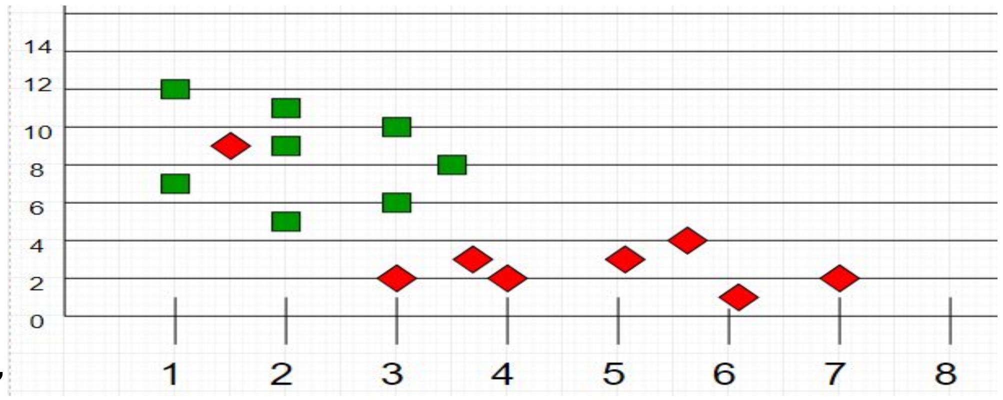


fig 3.4.1.2

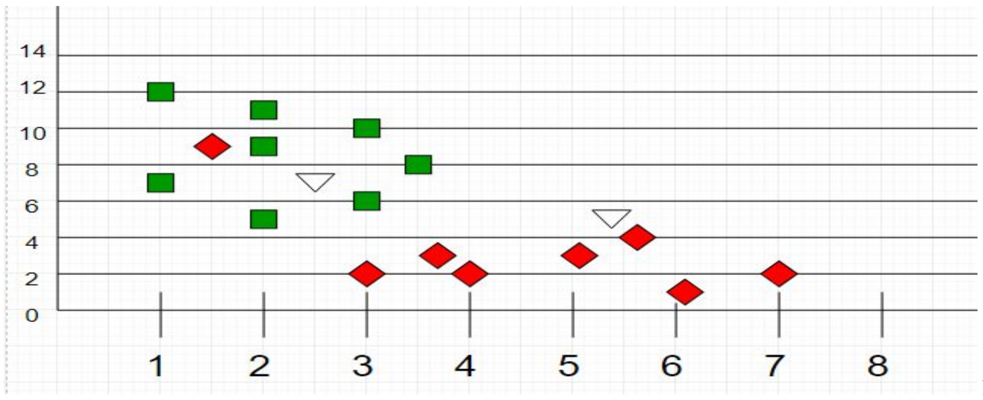


fig 3.4.1.3

The unclassified points are marked as White.

### 3.4.2 Gaussian Naive Bayes

Naive Bayes can be extended to real-valued attributes, most commonly by assuming a Gaussian distribution.This extension of naive Bayes is called Gaussian Naive Bayes. Other functions can be used to estimate the distribution of the data, but the Gaussian (or Normal distribution) is the easiest to work with because you only need to estimate the mean and the standard deviation from your training data.

**Representation for Gaussian Naive Bayes**

Above, we calculated the probabilities for input values for each class using a frequency. With real-valued inputs, we can calculate the mean and standard deviation of input values (x) for each class to summarize the distribution.This means that in addition to the probabilities for each class, we must also store the mean and standard deviations for each input variable for each class.

**Gaussian Naive Bayes Model From Data**

This is as simple as calculating the mean and standard deviation values of each input variable (x) for each class value.

$$\text{mean}(x) = 1/n * \text{sum}(x)$$

Where n is the number of instances and x are the values for an input variable in your training data.

We can calculate the standard deviation using the following equation:

$$\text{standard deviation}(x) = \text{sqrt}(1/n * \text{sum}(xi-\text{mean}(x)\hat{}2 ))$$

This is the square root of the average squared difference of each value of x from the mean value of x, where n is the number of instances, sqrt() is the square root function, sum() is the sum function, xi is a specific value of the x variable for the i'th instance and mean(x) is described above, and ^2 is the square.

### 3.4.3 Random Forest Classifier

Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision *trees*, resulting in a forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.

**How Random Forest Works:**

The following are the basic steps involved in performing the random forest algorithm

1. Pick N random records from the dataset.
2. Build a decision tree based on these N records.
3. Choose the number of trees you want in your algorithm and repeat steps 1 and 2.

For classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote

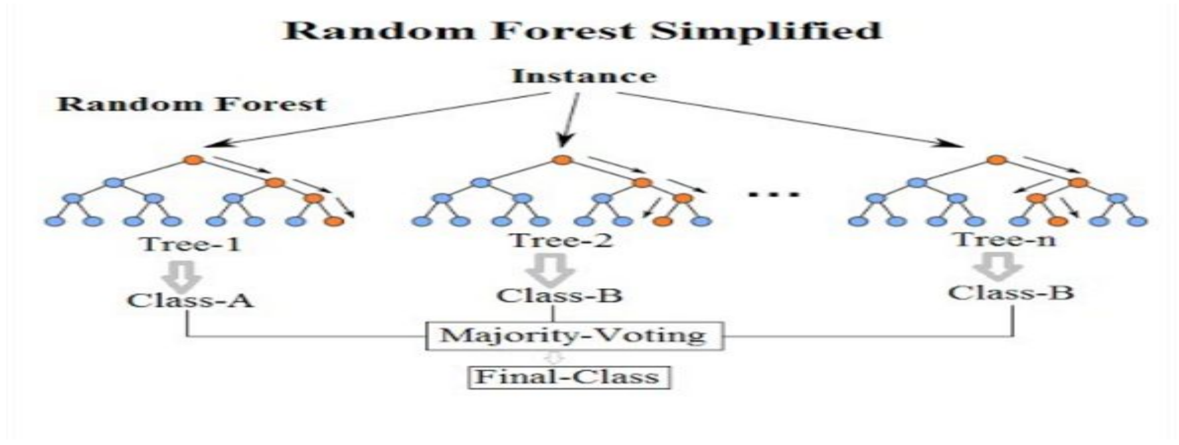


fig 3.4.3.1

## 3.4.4 Logistic regression

- Logistic regression is another technique borrowed by machine learning from the field of statistics.
- It is the go-to method for binary classification problems (problems with two class values). .
- It is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary).
- Like all regression analyses, the logistic regression is a predictive analysis.
- Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.
- The dependent variable should be dichotomous in nature (e.g., presence vs. absent).
- There should be no outliers in the data, which can be assessed by converting the continuous predictors to standardized scores, and removing values below -3.29 or greater than 3.29.

At the center of the logistic regression analysis is the task estimating the log odds of an event. Mathematically, logistic regression estimates a multiple linear regression function defined as:

$$\text{logit(p)} = \log\left(\frac{p(y=1)}{1-(p=1)}\right) = \beta_0 + \beta_1 \cdot x_{i2} + \beta_2 \cdot x_{i2} + \_ + \beta_p \cdot x_{in}$$

for i = 1…n .

**Overfitting.** When selecting the model for the logistic regression analysis, another important consideration is the model fit. Adding independent variables to a logistic regression model will always increase the amount of variance explained in the log odds (typically expressed as $R^2$). However, adding more and more variables to the model can result in overfitting, which reduces the generalizability of the model beyond the data on which the model is fit.

**Reporting the R2**. Numerous pseudo-R2 values have been developed for binary logistic regression. These should be interpreted with extreme caution as they have many computational issues which cause them to be artificially high or low. A better approach is to present any of the goodness of fit tests available; Hosmer-Lemeshow is a commonly used measure of goodness of fit based on the Chi-square test.

### 3.4.5 Gradient boosting Machine

Boosting is a method of converting weak learners into strong learners. In boosting, each new tree is a fit on a modified version of the original data set. The gradient boosting algorithm (gbm) can be most easily explained by first introducing the AdaBoost Algorithm.The AdaBoost Algorithm begins by training a decision tree in which each observation is assigned an equal weight. After evaluating the first tree, we increase the weights of those observations that are difficult to classify and lower the weights for those that are easy to classify. The second tree is therefore grown on this weighted data. Here, the idea is to improve upon the predictions of the first tree. Our new model is therefore *Tree 1 + Tree 2*. We then compute the classification error from this new 2-tree ensemble model and grow a third tree to predict the revised residuals. We repeat this process for a specified number of iterations. Subsequent trees help us to classify observations that are not well classified by the previous trees. Predictions of the final ensemble model is therefore the weighted sum of the predictions made by the previous tree models.

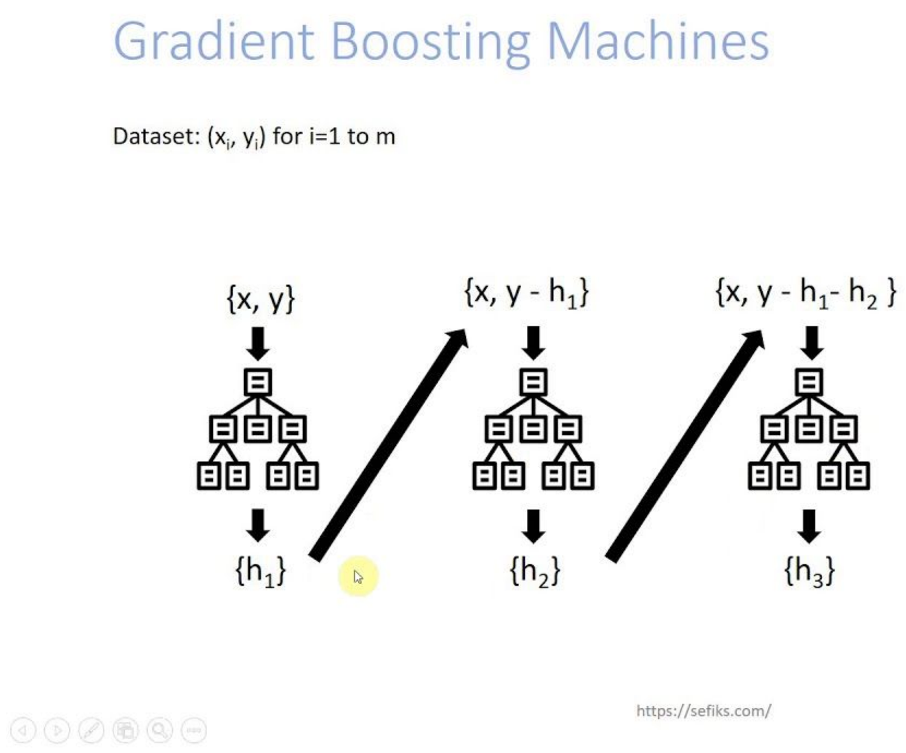


fig 3.4.5.1

### 3.4.6 SVM

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyperplane that differentiates the two classes very well.

Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier which best segregates the two classes (hyper-plane/ line)

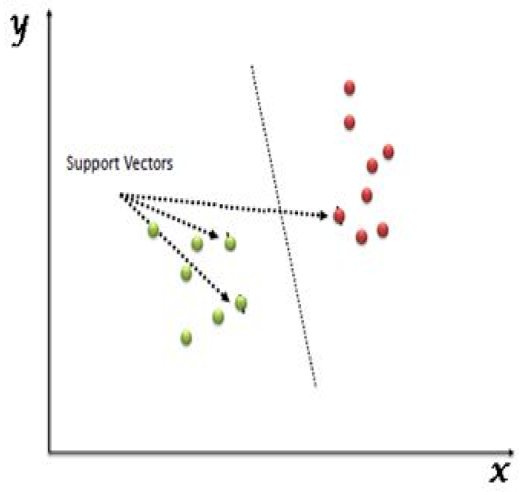


fig 3.4.6.1

### 3.4.7 XGBoost

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now. Please see the chart below for the evolution of tree-based algorithms over the years.

**Why does XGBoost perform so well?**

XGBoost and Gradient Boosting Machines (GBMs) are both ensemble tree methods that apply the principle of boosting weak learners (CARTs generally) using the gradient descent architecture. However, XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements.

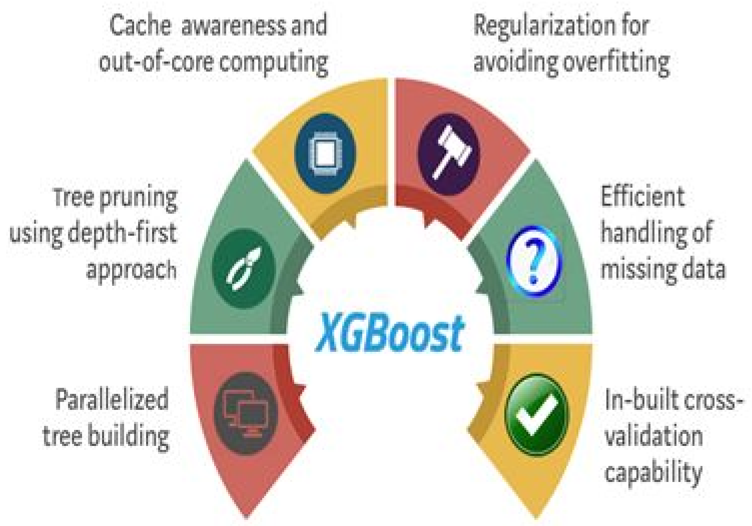

Cache awareness and out-of-core computing
Regularization for avoiding overfitting
Tree pruning using depth-first approach
Efficient handling of missing data
XGBoost
Parallelized tree building
In-built cross-validation capability

fig 3.4.7.1

# 4. DESIGN

## 4.1 UML diagrams

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. The UML addresses the documentation of a system's architecture and all of its details. The UML also provides a language for expressing requirements and for modeling the activities of project planning.

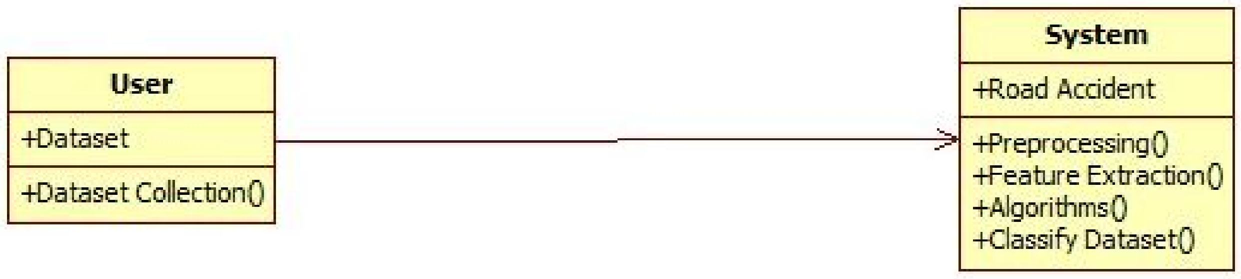
Building blocks of UML:

UML encompasses three kinds of building blocks:

- Things
- Relationships
- Diagrams

**4.1.1 Class Diagram:**Class diagram is a static diagram. It represents the static view of an application.Class diagram describes the attributes and operations of a class and also the constraints imposed on the system.Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

fig 4.1.1.2



### 4.1.2 Sequence Diagram:

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function

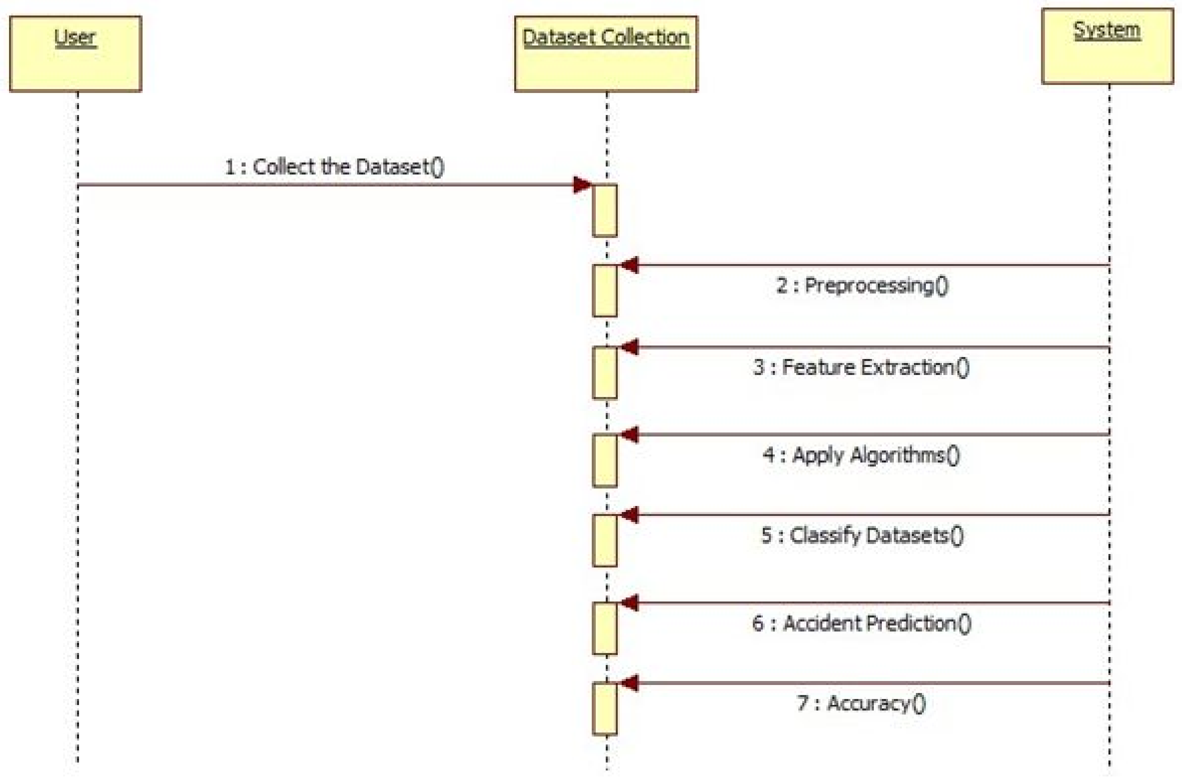


fig 4.1.2.1

### 4.1.3 Activity Diagram:

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc

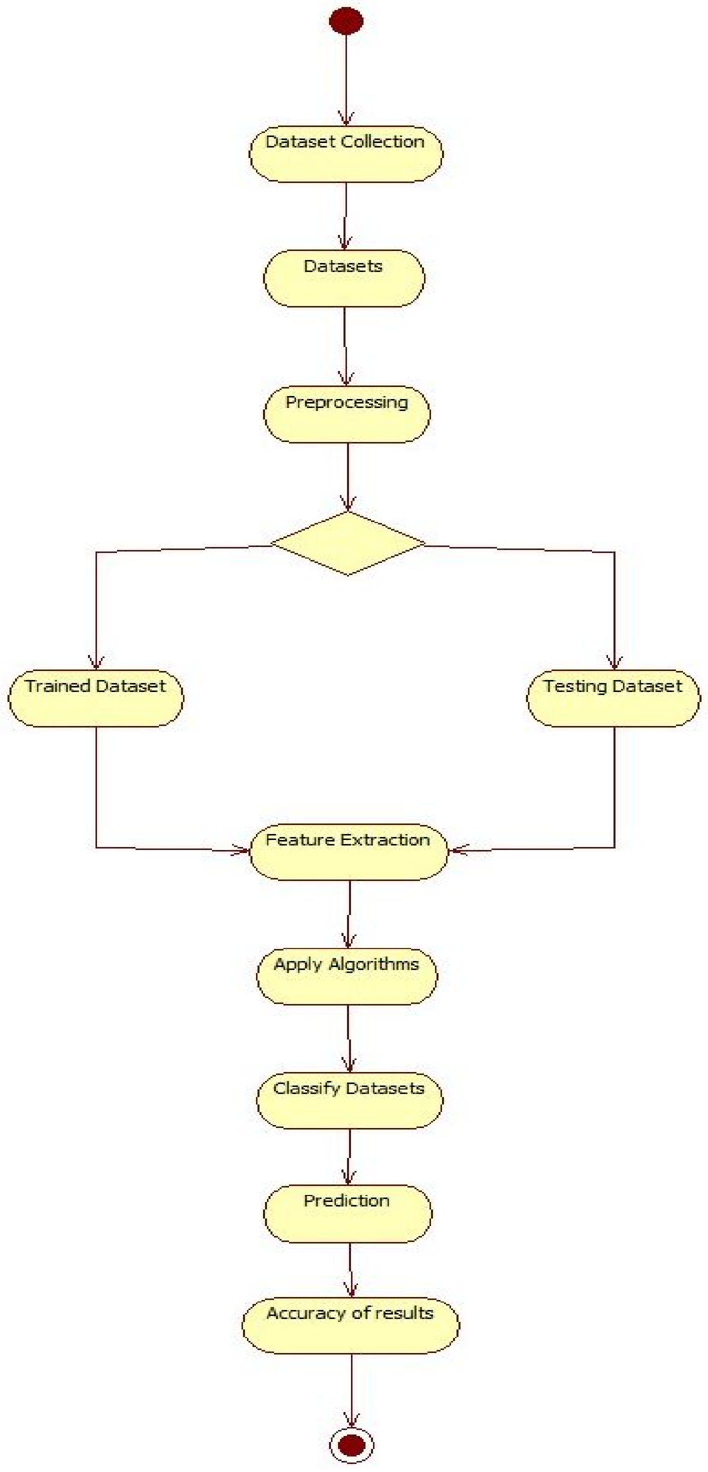
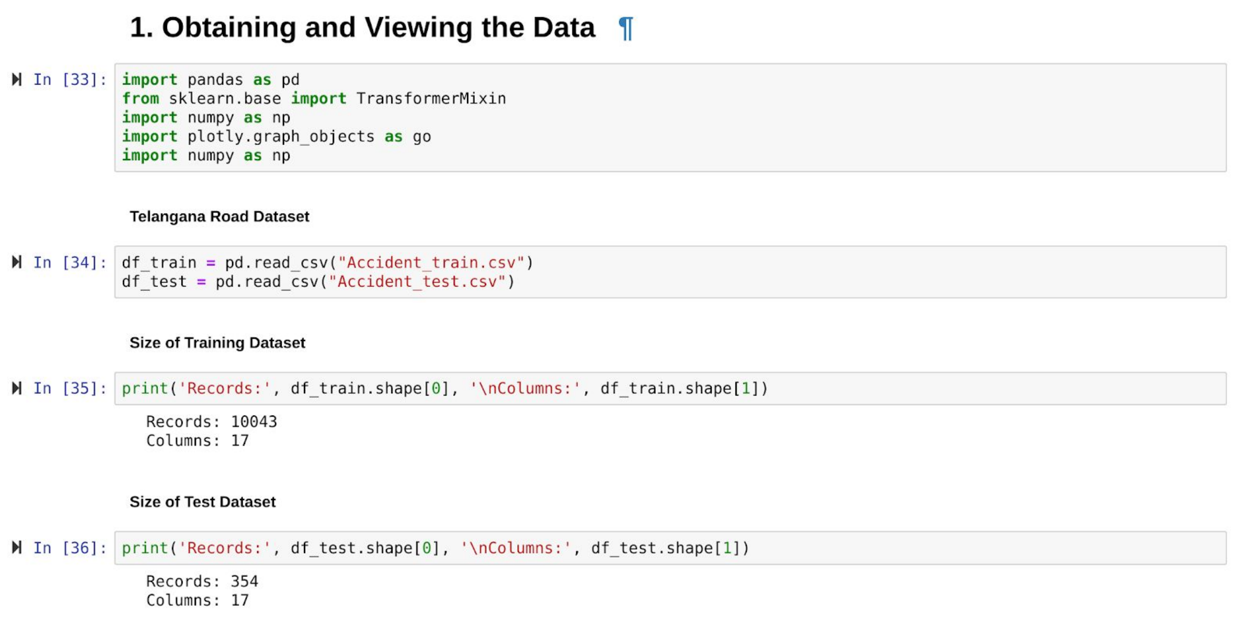

Dataset Collection
Datasets
Preprocessing
Trained Dataset
Testing Dataset
Feature Extraction
Apply Algorithms
Classify Datasets
Prediction
Accuracy of results

fig 4.1.3.1

# 5. Implementation:

## 5.1 Code

### 5.1.1 Obtaining and viewing the data



## 1. Obtaining and Viewing the Data ¶

In [33]:
```
import pandas as pd
from sklearn.base import TransformerMixin
import numpy as np
import plotly.graph_objects as go
import numpy as np
```

**Telangana Road Dataset**

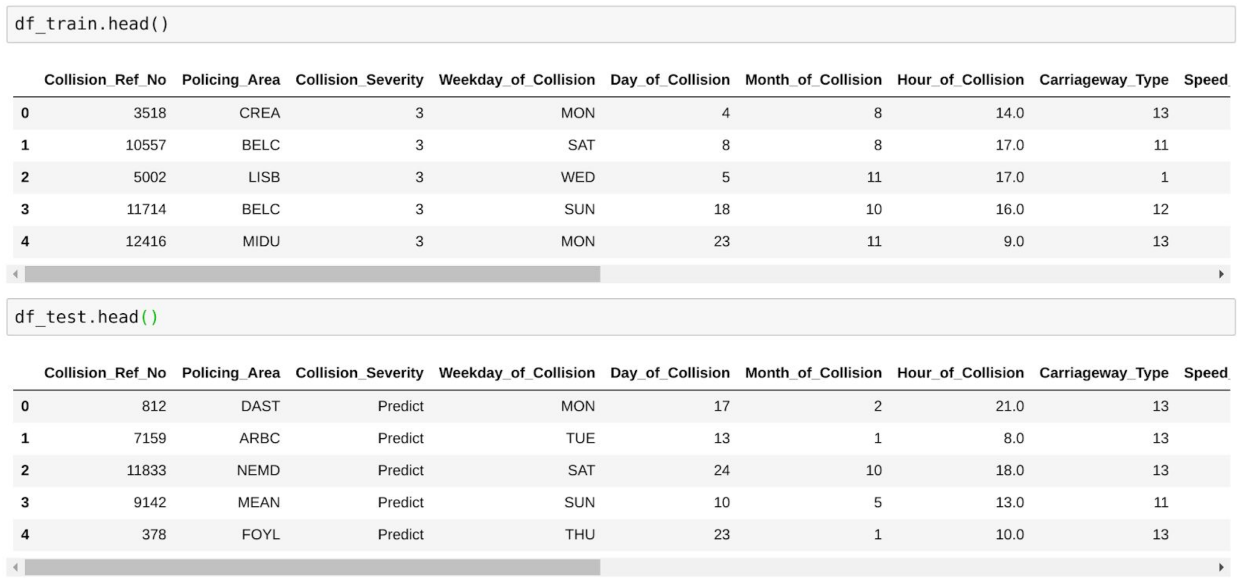
In [34]:
```
df_train = pd.read_csv("Accident_train.csv")
df_test = pd.read_csv("Accident_test.csv")
```

**Size of Training Dataset**

In [35]:
```
print('Records:', df_train.shape[0], '\nColumns:', df_train.shape[1])
```
```
Records: 10043
Columns: 17
```

**Size of Test Dataset**

In [36]:
```
print('Records:', df_test.shape[0], '\nColumns:', df_test.shape[1])
```
```
Records: 354
Columns: 17
```



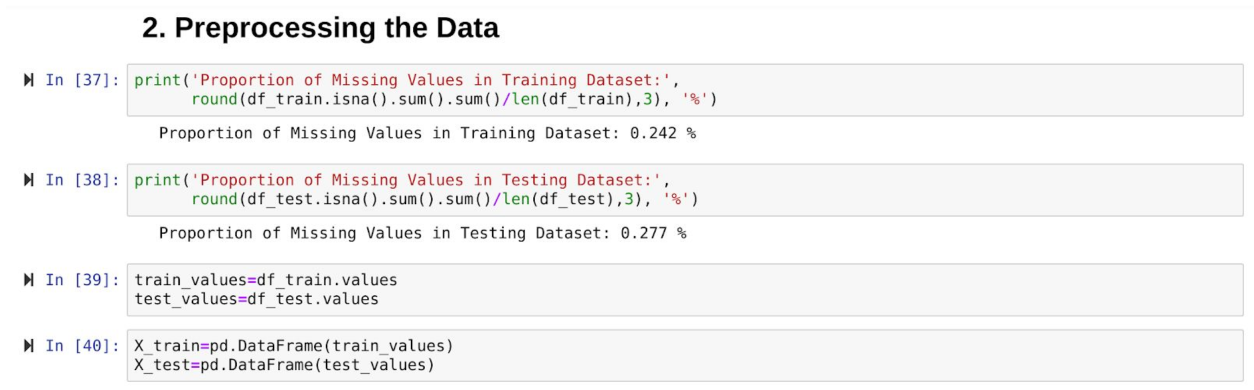
```
df_train.head()
```

| | Collision_Ref_No | Policing_Area | Collision_Severity | Weekday_of_Collision | Day_of_Collision | Month_of_Collision | Hour_of_Collision | Carriageway_Type | Speed_ |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 3518 | CREA | 3 | MON | 4 | 8 | 14.0 | 13 | |
| **1** | 10557 | BELC | 3 | SAT | 8 | 8 | 17.0 | 11 | |
| **2** | 5002 | LISB | 3 | WED | 5 | 11 | 17.0 | 1 | |
| **3** | 11714 | BELC | 3 | SUN | 18 | 10 | 16.0 | 12 | |
| **4** | 12416 | MIDU | 3 | MON | 23 | 11 | 9.0 | 13 | |

```
df_test.head()
```

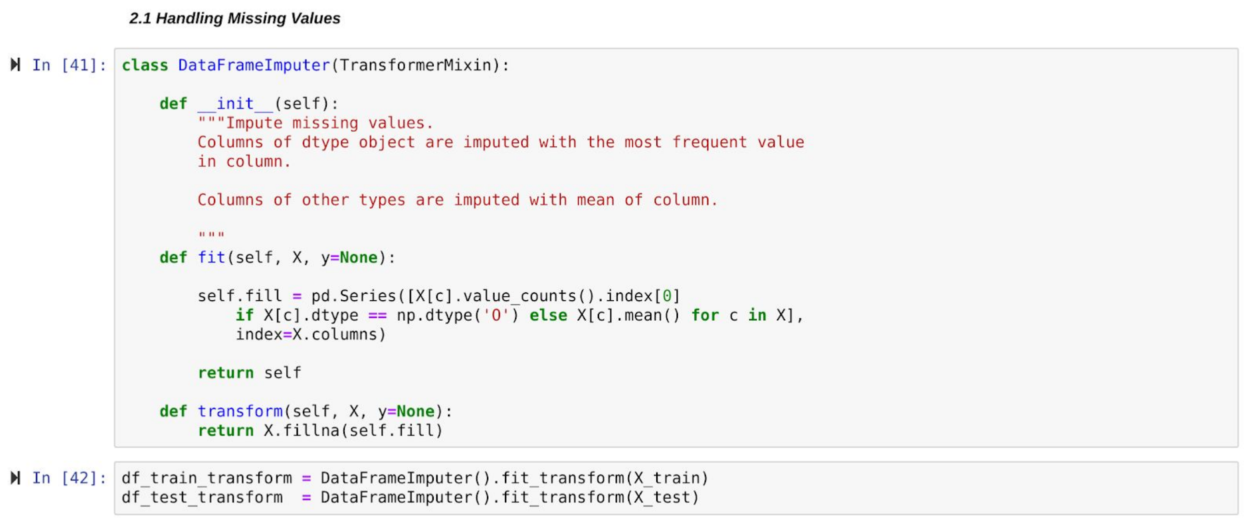
| | Collision_Ref_No | Policing_Area | Collision_Severity | Weekday_of_Collision | Day_of_Collision | Month_of_Collision | Hour_of_Collision | Carriageway_Type | Speed_ |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 812 | DAST | Predict | MON | 17 | 2 | 21.0 | 13 | |
| **1** | 7159 | ARBC | Predict | TUE | 13 | 1 | 8.0 | 13 | |
| **2** | 11833 | NEMD | Predict | SAT | 24 | 10 | 18.0 | 13 | |
| **3** | 9142 | MEAN | Predict | SUN | 10 | 5 | 13.0 | 11 | |
| **4** | 378 | FOYL | Predict | THU | 23 | 1 | 10.0 | 13 | |

## 5.1.2 Preprocessing the Data



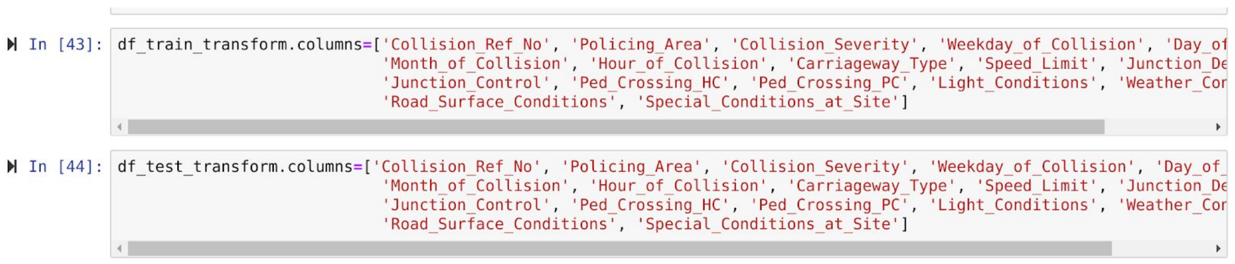
## 2. Preprocessing the Data

```
In [37]: print('Proportion of Missing Values in Training Dataset:',
               round(df_train.isna().sum().sum()/len(df_train),3), '%')

            Proportion of Missing Values in Training Dataset: 0.242 %

In [38]: print('Proportion of Missing Values in Testing Dataset:',
               round(df_test.isna().sum().sum()/len(df_test),3), '%')

            Proportion of Missing Values in Testing Dataset: 0.277 %

In [39]: train_values=df_train.values
         test_values=df_test.values

In [40]: X_train=pd.DataFrame(train_values)
         X_test=pd.DataFrame(test_values)
```



*2.1 Handling Missing Values*

```
In [41]: class DataFrameImputer(TransformerMixin):

             def __init__(self):
                 """Impute missing values.
                 Columns of dtype object are imputed with the most frequent value
                 in column.

                 Columns of other types are imputed with mean of column.

                 """
             def fit(self, X, y=None):

                 self.fill = pd.Series([X[c].value_counts().index[0]
                     if X[c].dtype == np.dtype('O') else X[c].mean() for c in X],
                     index=X.columns)

                 return self

             def transform(self, X, y=None):
                 return X.fillna(self.fill)

In [42]: df_train_transform = DataFrameImputer().fit_transform(X_train)
         df_test_transform  = DataFrameImputer().fit_transform(X_test)
```



```
In [43]: df_train_transform.columns=['Collision_Ref_No', 'Policing_Area', 'Collision_Severity', 'Weekday_of_Collision', 'Day_of
                                     'Month_of_Collision', 'Hour_of_Collision', 'Carriageway_Type', 'Speed_Limit', 'Junction_De
                                     'Junction_Control', 'Ped_Crossing_HC', 'Ped_Crossing_PC', 'Light_Conditions', 'Weather_Cor
                                     'Road_Surface_Conditions', 'Special_Conditions_at_Site']

In [44]: df_test_transform.columns=['Collision_Ref_No', 'Policing_Area', 'Collision_Severity', 'Weekday_of_Collision', 'Day_of_
                                    'Month_of_Collision', 'Hour_of_Collision', 'Carriageway_Type', 'Speed_Limit', 'Junction_De
                                    'Junction_Control', 'Ped_Crossing_HC', 'Ped_Crossing_PC', 'Light_Conditions', 'Weather_Cor
                                    'Road_Surface_Conditions', 'Special_Conditions_at_Site']
```

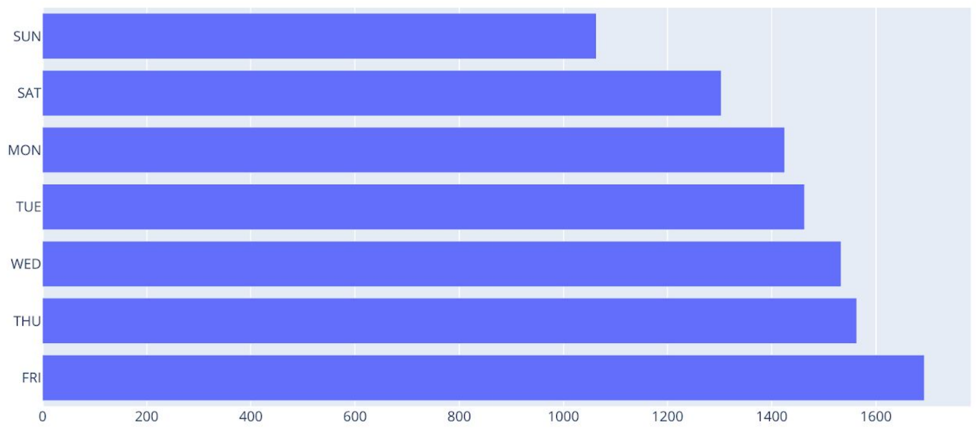
**5.1.3**

## 3. Exploratory Data Analysis (EDA)

***On which major factors does accidents most likely to be caused?***

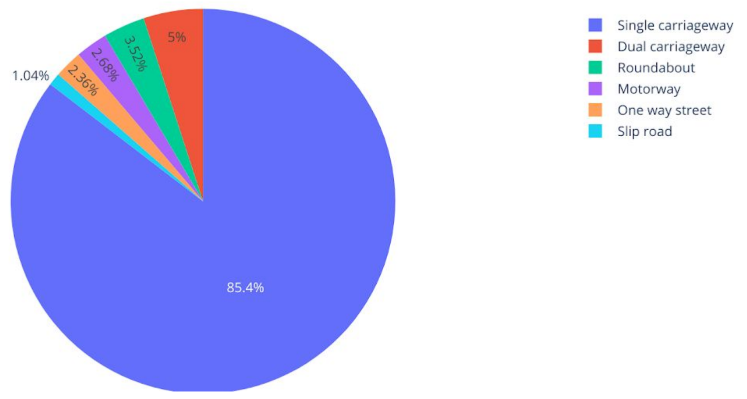
***3.1 Preparing dataframe that calculates accidents per weekday:***

```
In [45]: week_day_cnts = df_train_transform['Weekday_of_Collision'].value_counts()
         groupby_week_day = week_day_cnts.to_dict()
         labels = list(groupby_week_day.keys())
         values = list(groupby_week_day.values())
         fig = go.Figure(go.Bar(x=values, y=labels, orientation='h'))
         fig.show()
```



***3.2 Road type when accident happened***

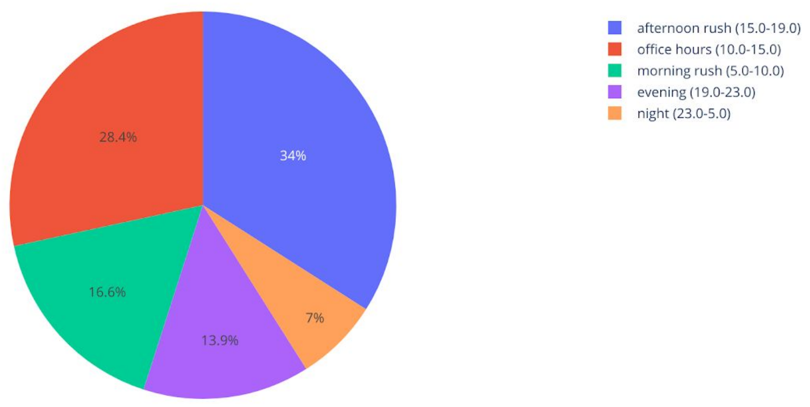
```
In [46]: carriageway_types = {1 : 'Roundabout', 2: 'One way street', 10: 'Other / unknown', 11: 'Dual carriageway', 12: 'Motorw
         ct = df_train_transform['Carriageway_Type'].value_counts()
         groupby_carriageway_types = ct.to_dict()
         labels = list(groupby_carriageway_types.keys())
         final_labels = [carriageway_types[i] for i in labels]
         values = list(groupby_carriageway_types.values())
         fig = go.Figure(data=[go.Pie(labels=final_labels, values=values)])
         fig.show()
```



***3.3 Accidents categorized by hour of collision***

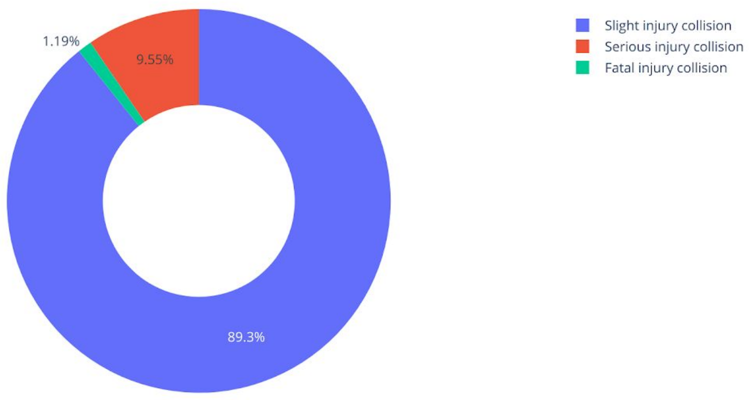
```
In [47]: def when_was_it(hour):
             if hour >= 5.0 and hour < 10.0:
                 return "morning rush (5.0-10.0)"
             elif hour >= 10.0 and hour < 15.0:
                 return "office hours (10.0-15.0)"
             elif hour >= 15.0 and hour < 19.0:
                 return "afternoon rush (15.0-19.0)"
             elif hour >= 19.0 and hour < 23.0:
                 return "evening (19.0-23.0)"
             else:
                 return "night (23.0-5.0)"
```

In [48]:
```
hr_of_col = df_train_transform['Hour_of_Collision'].value_counts()
groupby_hr_col = hr_of_col.to_dict()
labels = list(groupby_hr_col.keys())
final_labels = [when_was_it(i) for i in labels]
values = list(groupby_hr_col.values())
fig = go.Figure(data=[go.Pie(labels=final_labels, values=values)])
fig.show()
```



***3.5 Percentage of each category of accident severity***

In [50]:
```
col_sev_types = {1 : 'Fatal injury collision', 2: 'Serious injury collision', 3: 'Slight injury collision'}
col_sev = df_train_transform['Collision_Severity'].value_counts()
groupby_col_sev = col_sev.to_dict()
labels = list(groupby_col_sev.keys())
final_labels = [col_sev_types[i] for i in labels]
values = list(groupby_col_sev.values())
fig = go.Figure(data=[go.Pie(labels=final_labels, values=values, hole=.5)])
fig.show()
```



**Encoding weekdays into numeric form**

```
In [51]: week_days = {'FRI': 5, 'MON': 1, 'SAT': 6, 'SUN': 7, 'THU': 4, 'TUE': 2, 'WED': 3}
         df_train_transform['Weekday_of_Collision'] = [week_days[day] for day in df_train_transform['Weekday_of_Collision']]
```

***Defining features as X and y as class labels***

```
In [52]: y = df_train_transform['Collision_Severity']
         X = df_train_transform.drop(['Collision_Severity', 'Policing_Area'], axis=1)
```

```
In [53]: print(len(X.columns))
```

```
15
```

## Define a cross validation strategy

I use the cross_val_score function of Sklearn. However this function has no shuffle attribute, I add one line of code, in order to shuffle the dataset prior to cross-validation.

For the performance metric, as I mentioned in the beginning, this is an imbalanced dataset. Thus, instead of using accuracy, I used F1-score as metric. F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

```
In [54]: from sklearn.metrics import make_scorer, f1_score
         from sklearn.model_selection import KFold,cross_val_score
         n_folds = 5
         def f1_cv(model):
             kf = KFold(n_folds, shuffle = True, random_state = 29).get_n_splits(X)
             scorer = make_scorer(f1_score, average = 'weighted')
             f1 = cross_val_score(model, X, y, scoring = scorer, cv = kf)
             return (f1)
```

### 5.1.4

## 4. Implementing Machine Learning algorithms

***4.1 KNN***

***4.2 Naive bayes classifier***

***4.3 Random forest classifier***

***4.4 Logistic regression***

***4.5 Gradient boosting Machine***

***4.6 SVM***

***4.7 XGBoost***

```
In [56]: from sklearn.neighbors import KNeighborsClassifier
         knn = KNeighborsClassifier(n_neighbors=3,weights='distance')
```

```
In [57]: from sklearn.naive_bayes import GaussianNB
         gnb = GaussianNB()
```

```
In [58]: from sklearn.ensemble import RandomForestClassifier
         rfc = RandomForestClassifier(n_estimators = 31, random_state = 32)
```

```
In [59]: from sklearn.linear_model import LogisticRegression
         lg = LogisticRegression( max_iter=5000,penalty='l2', n_jobs=3,
                                  solver='lbfgs', verbose=1)
```

```
In [60]: from sklearn.ensemble import GradientBoostingClassifier
         GBoost = GradientBoostingClassifier( n_estimators=3000, learning_rate=0.05,
                                             max_depth=4, max_features='sqrt',
                                             min_samples_leaf=15, min_samples_split=10,
                                             random_state =5)
```

```
In [61]: from sklearn.svm import SVC
         svc = SVC(kernel = 'sigmoid', gamma = 1.0)
```

```
In [62]: from xgboost import XGBClassifier
         xgboost = XGBClassifier(learning_rate =0.07, n_estimators=300,
                                 class_weight="balanced_subsample",
                                 max_depth=8, min_child_weight=1,
                                 scale_pos_weight=7,
                                 seed=27,subsample=0.8,colsample_bytree=0.8)
```

**Predicting the Mean and Standard Deviation of each classifier** ¶

```
In [63]: score = f1_cv(svc)
         print ('\nSVC score: {:4f}({:4f})\n'.format(score.mean(), score.std()))

            SVC score: 0.841893(0.000266)
```

```
In [64]: score = f1_cv(gnb)
         print ('gnb score: {:4f}({:4f})\n'.format(score.mean(), score.std()))
            gnb score: 0.835021(0.007351)
```

```
In [65]: score = f1_cv(rfc)
         print ('rfc score: {:4f}({:4f})\n'.format(score.mean(), score.std()))
            rfc score: 0.858807(0.004937)
```

```
In [66]: score = f1_cv(lg)
         print ('lg score: {:4f}({:4f})\n'.format(score.mean(), score.std()))
            [Parallel(n_jobs=3)]: Done   1 out of   1 | elapsed:   12.5s finished
            [Parallel(n_jobs=3)]: Done   1 out of   1 | elapsed:   11.6s finished
            [Parallel(n_jobs=3)]: Done   1 out of   1 | elapsed:    4.5s finished
            [Parallel(n_jobs=3)]: Done   1 out of   1 | elapsed:    6.7s finished
            lg score: 0.841843(0.000303)

            [Parallel(n_jobs=3)]: Done   1 out of   1 | elapsed:    6.0s finished
```

```
In [69]: score = f1_cv(knn)
         print ('\nKNN score: {:4f}({:4f})\n'.format(score.mean(), score.std()))
```

```
KNN score: 0.831996(0.003223)
```

**Spliting the train data into training set and test set in 70-30 ratio**

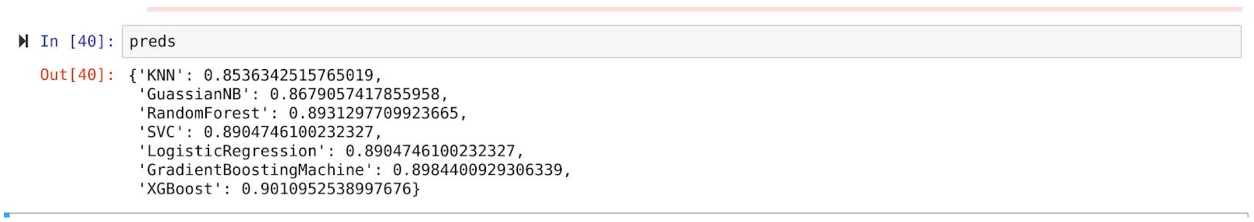
```
In [71]: from sklearn.model_selection import train_test_split

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
In [72]: def classifier(clf, X_train, y_train):
             clf.fit(X_train, y_train)
         def predictor(clf, X_test):
             return (clf.predict(X_test))
```

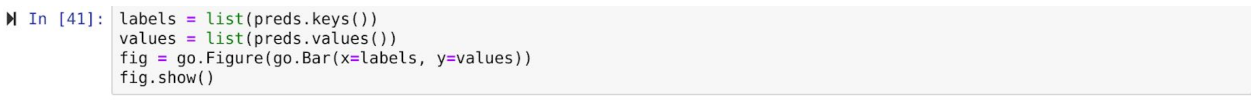
```
In [73]: from sklearn.metrics import accuracy_score
         clf = {'KNN': knn, 'GuassianNB':gnb, 'RandomForest':rfc,'SVC':svc,'LogisticRegression': lg, 'GradientBoostingMachine':
         preds = {}
         for key, value in clf.items():
             classifier(value, X_train, y_train)
             pred = predictor(value,X_test)
             preds[key] = accuracy_score(y_test,pred)
```

```
[Parallel(n_jobs=3)]: Done   1 out of   1 | elapsed:    6.2s finished
```

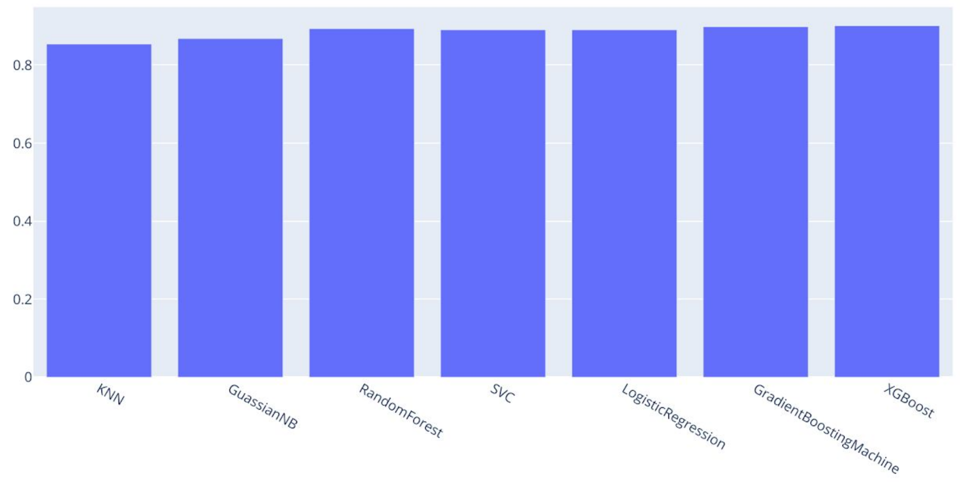
## 5.2 Output Screens



```
In [40]: preds
Out[40]: {'KNN': 0.8536342515765019,
          'GuassianNB': 0.8679057417855958,
          'RandomForest': 0.8931297709923665,
          'SVC': 0.8904746100232327,
          'LogisticRegression': 0.8904746100232327,
          'GradientBoostingMachine': 0.8984400929306339,
          'XGBoost': 0.9010952538997676}
```



```
In [41]: labels = list(preds.keys())
         values = list(preds.values())
         fig = go.Figure(go.Bar(x=labels, y=values))
         fig.show()
```



# 6. Test Cases

Software testing is the process used to help identify the correctness, completeness, security, and quality of developed computer software. Testing is a process of technical investigation, performed on behalf of stakeholders, that is intended to reveal quality related information about the product with respect to the context in which it is intended to operate.

## 6.1 Unit Testing

Unit testing, a testing technique using which individual modules are tested to determine if there are any issues by the developer himself/herself. It is concerned with functional correctness of standalone modules. Unit testing is a component of test driven development (TTD), a pragmatic methodology that takes a meticulous approach to building a product by means of continual testing and revision. Test driven development requires that developers first write failing unit tests. Then, they write code and re factor the application until the test passes. TTD typically results in an explicit and predictable code base.

Unit testing involves only those characteristics that are vital to the performance of the unit under test. This encourages developers to modify the source code without immediate concerns about how such changes might affect the functioning of other units or the program as a whole. Once all of the units in a program have been found to be working in the most efficient and error free manner possible, larger components of the program can be evaluated by means of integration testing.

Unit testing does have a learning curve. The development team needs to learn what unit testing is, how to unit test, what to unit test and how to use automated software tools to facilitate the process on an ongoing basis. The great benefit to unit testing is that the earlier the problem is identified, the fewer compound errors occur. A compound error is one that doesn't seem to break anything at first, but eventually conflicts with something down the line and results in a problem.

The main aim is to isolate each unit of the system to identify, analyze and fix the defects.

Advantages :

- Reduces the defects in the newly developed features or reduces the bugs when changing the existing functionality.

- Reduces cost of testing as defects are captured in very early phase.
- Improves design and allows better refactoring of code.
- Unit tests, when integrated with build gives the quality of build as well.

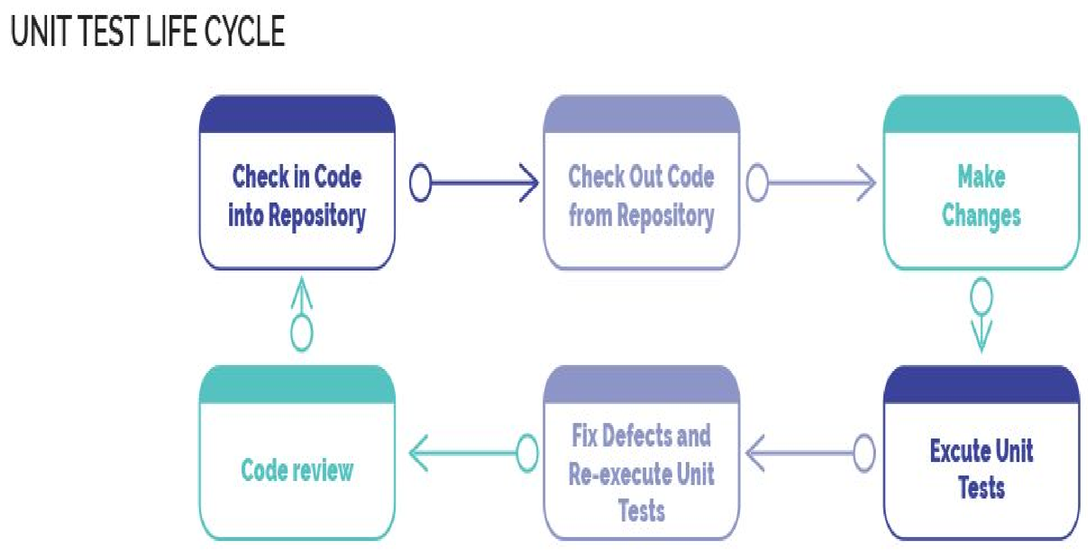

UNIT TEST LIFE CYCLE
Check in Code into Repository
Check Out Code from Repository
Make Changes
Excute Unit Tests
Fix Defects and Re-execute Unit Tests
Code review

**Figure 6.1.1: Unit Testing life cycle**

## 6.2 Unit Test Cases

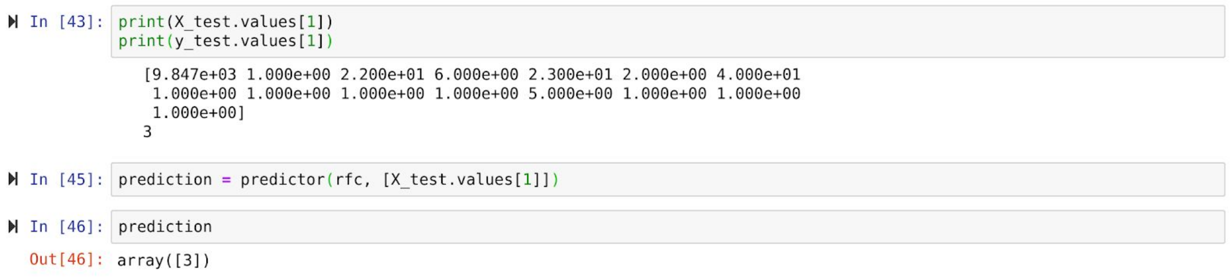
The Telangana road severity dataset  has three class labels.The basic testing that is unit testing is done over the model by passing features as arguments over the cell in jupyter notebook and checking the results with  expected outputs. The results were so pleasing that there was not much differences between predicted and expected outputs. As a result of observation, we finally simulated that the accident severity model efficiently outputs whether there could be change for accident.

The following are few test cases where we gave input as an array of features.

**Table 6.2.1: Unit test cases**

| S. No | Input | Expected Outcome | Actual Outcome |
|---|---|---|---|
| 1 | [9.847e+03 1.000e+00 2.200e+01 6.000e+00 2.300e+01 2.000e+00 4.000e+01 1.000e+00 1.000e+00 1.000e+00 1.000e+00 5.000e+00 1.000e+00 1.000e+00 1.000e+00] | 3 | 3 |
| 2 | [4.045e+03 1.000e+00 8.000e+00 9.000e+00 1.600e+01 1.300e+01 6.000e+01 1.000e+00 7.000e+00 1.000e+00 1.000e+00 2.000e+00 1.000e+00 1.000e+00 1.000e+00] | 2 | 2 |
| 3 | [1.1089e+04 4.0000e+00 1.0000e+01 9.0000e+00 2.0000e+01 1.3000e+01 3.0000e+01 1.2000e+01 4.0000e+00 1.0000e+00 1.0000e+00 4.0000e+00 1.0000e+01 1.0000e+00 1.0000e+00] | 3 | 2 |

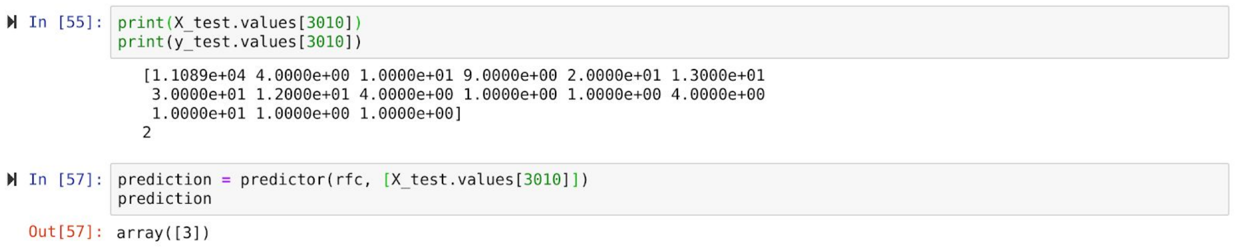
**Figure 6.2.2: Unit Test Case - I**



```
In [43]: print(X_test.values[1])
         print(y_test.values[1])

         [9.847e+03 1.000e+00 2.200e+01 6.000e+00 2.300e+01 2.000e+00 4.000e+01
          1.000e+00 1.000e+00 1.000e+00 1.000e+00 5.000e+00 1.000e+00 1.000e+00
          1.000e+00]
         3

In [45]: prediction = predictor(rfc, [X_test.values[1]])

In [46]: prediction
Out[46]: array([3])
```

**Figure 6.2.3: Unit Test Case - 2**

**Figure 6.2.4: Unit Test Case - 3**



```
In [55]: print(X_test.values[3010])
         print(y_test.values[3010])

         [1.1089e+04 4.0000e+00 1.0000e+01 9.0000e+00 2.0000e+01 1.3000e+01
          3.0000e+01 1.2000e+01 4.0000e+00 1.0000e+00 1.0000e+00 4.0000e+00
          1.0000e+01 1.0000e+00 1.0000e+00]
         2

In [57]: prediction = predictor(rfc, [X_test.values[3010]])
         prediction
Out[57]: array([3])
```

# 7. Conclusion and Future Scope

Our main aim was to predict the severity of the accident when it is “serious” and “fatal”. Using the Telangana Dataset we were able to run most of our algorithms. Data is highly imbalanced so even though most of our algorithms were giving > 89% accuracies, it was of no use. It was predicting all the accidents as slight accidents. After checking on all these algorithms, the team even tried dimensionality reduction techniques, but the results were not improved. Then the team decided to use the undersampled dataset as it was giving better results in predicting the severe/fatal accidents. This decision was made on trying out oversampling, undersampling, test and train data with an equal ratio of classification classes.

**Scope for Future enhancement**

In conclusion, most of the algorithms are biased towards the most frequent class. However, efficient pre-processing and corresponding imbalanced data techniques should give optimal results. Based on the current known conditions of weather, light, traffic signal, road surface, speed limit etc., accident severity can be classified. But there is no one feature that influences the accident severity. Future work involves considering regions from latitude and longitude and this problem can be turned into a regression problem. We can then predict the risk of accidents in the given region. If the risk is higher then immediate actions can be taken.

## 8. References

[1] Comparison of Machine learning Algorithms for predicting TrafficAccident Severity,2019

[2] Traffic Accident Severity prediction using Naive Bayes Algorithm ACISE,2019

[3] Influencing attributes for traffic accident severity prediction model, ICTHC 2019

[4] Prediction of Accident Severity Based on Random Forest and LogisticRegression Model,2019

[5] Road Accident Analysis and Prediction of Accident Severity by Using Machine Learning in Bangladesh,2019.

[6] Traffic Accident Classification and Injury Severity Prediction, June 2019

[7] Prediction for traffic accident severity:comparing bayesian network and regression Regression model,2019.

[8] Road accidents analysis and prediction by using Machine learning,2019.