A MINI PROJECT REPORT ON
"PREDICTION OF ACCIDENT INJURY SEVERITY USING NEURAL NETWORKS"

Submitted in partial fulfilment of the requirement
for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

By:

Tanvi Koduru 17P61A05A2
Ajay Ganji 17P61A0561
Keerthi Katam 17P61A0597
Vinay Kumar 17P61A0565

Under the guidance of
Mr. Praveen Kumar
Associate Professor

VIGNANA BHARATHI INSTITUTE OF TECHNOLOGY
(A UGC Autonomous Institution, Approved by AICTE, Affiliated to JNTUH, Kukatpally
Accredited by National Board of Accreditation (NBA),
National Assessment and Accreditation Council (NAAC))
Aushapur (V), Ghatkesar (M), Medchal(dist.).

2020-2021

(A UGC Autonomous Institution, Approved by AICTE, Affiliated to JNTUH, Kukatpally
Accredited by National Board of Accreditation (NBA),
National Assessment and Accreditation Council (NAAC))

CERTIFICATE

This is to certify that the project report entitled "Prediction Of Accident Injury Severity" submitted by Tanvi Koduru, Keerthi Katam, Ajay Ganji, Vinay Kumar and carried out under the guidance of Mr. Praveen Kumar - Associate Prof, Computer Science And Engineering, Vignana Bharathi Institute Of Technology is a record of their own work. The matter present in this thesis is original and has not been submitted to any other university for the award of any other degree.

INTERNAL GUIDE	**HEAD OF THE DEPT.**	**EXTERNAL EXAMINER**
Mr. Praveen Kumar	Mr. N. Srinivas	
(Associate Prof.)		

DATE :

DECLARATION

We hereby declare that the project entitled “Predicting Severity Of Accident Injuries” being submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science And Engineering is the authentic record of our own work done under the guidance Of **Mr. Praveen Kumar.**

We also declare that this project work has neither been submitted to any other board not published at any time by us in the past.

Tanvi Koduru
Ajay Ganji
Keerthi Katam
Vinay Kumar

ACKNOWLEDGEMENT

In this competitive world, it has become a difficult task for many of us to run the race. Not every one of us is able to connect theoretical learning with practical working. There has always been a gap between the two and we could fill this gap. All thanks to our faculty and the management for helping us bridge this gap.

We also would like to thank Mr. Srinivas, Head of the Department, Computer Science and Engineering for the extreme support throughout the doing of our project.

We, specially would like to thank our project guide Mr. Praveen Kumar, Associate Professor, for being the pillar of support and by helping us in successful completion of this project.

We also would like to express our sincere thanks to our project coordinators Mr. Hanumanth Rao and Mrs. Swapna for guiding and suggesting us whenever and wherever required.

TABLE OF CONTENTS

1. ABSTRACT

Prediction of accident severity is one of the most pivotal steps in the accident management process. Ideal recognition of the influential factors may play a significant job in traffic safety. In this project, we are taking a shot at foreseeing the seriousness of injuries brought about by a mishap.

In this methodology, we remove the prevailing insights identified with the vehicle crashes and store them in a dataset. In this task we will utilize a progression of neural networks to show the possibly non-straight connections between the injury seriousness levels and crash-related elements. We at that point attempt to anticipate the degree of seriousness of the injury.

2. INTRODUCTION

Traffic accidents are a primary concern due to many fatalities and economic losses every year worldwide. A need to have better accident injury severity prediction models has become very important for enhancing the safety performance of the road traffic system.

The factors that affect the accident severity are mainly related to the vehicle characteristics, speed of the vehicle in motion, driver characteristics, conditions of the road and atmospheric characteristics.

Using deep neural networks is a successful procedure that can be utilized to see the degree of seriousness of a mishap. This is done by the conditions under which drivers and travelers are bound to be executed or all the more seriously harmed in an accident can help improve the general driving security circumstance.

A few parts that are considered as those that impact the peril of extended injury of the occupants in an incident incorporate behavioral characteristics of an individual, natural components at the hour of the difficulty, particular issues with the vehicle, recklessness of people, alcohol usage, tranquilize involvement and numerous others. We split this information into training and testing informational collection and afterward verify the performance of the model delivered utilizing the test informational index.

In this undertaking we use one of the notable order strategies which is the neural system model to produce the characterization model. The consequence of this system lets us know how serious the wounds are caused after a mishap.

2.1 Problem Statement

1. The number of car accidents and their casualties has been a rising pattern all around the world because of the increase in population.

2. Various components engaged with car accidents substantially affect one another, accordingly making it hard to independently consider any of the boundaries while predicting the seriousness of injuries caused by accidents.

2.2 Objective

The main objectives of this model are :

- To perform parameter selection on the dataset by using a few methods to understand the most important and necessary parameters for building the model and also the not so important or necessary features in the process of prediction of the severity of injuries.

- To use these selected features and build a model by using neural networks.

- To predict the level of seriousness of an injury caused by an accident.

3.METHODOLOGY

Artificial Intelligence and Machine Learning have become the buzz words these days and are the cornerstones of the next revolution in computing. On the other hand Deep Learning is a subset of Machine Learning which is a field that is based on learning and improving on its own by examining computer algorithms.

In this project, we have used a deep neural network to train our model and build a classifier in order to predict the severity of an accident injury. The methodology and techniques that have been used to study the traffic accident dataset is described below:

3.1 Data Collection :

After a proper research we have gathered a dataset of accident records which was obtained from the United States with 30 different attributes. The total number of accident records that we have used are 100969. The different attributes for each accident record that have been taken into consideration are: Case State, Age, Gender, Person Type, Seating Position, Restraint System Use, Air Bag Availability/ Deployment, Ejection, Ejection Path, Extrication , Non-Motorist Location, Police Reported Alcohol Involvement, Method Alcohol Determination, Alcohol Test Type, Alcohol Test Result, Police Reported Drug Involvement, Method of Drug Determination, Drug Test Type, Drug Test Results(1 of 3), Drug Test Type(2 of 3), Drug Test Result(2 of 3), Drug Test Type(3 of 3), Drug Test Results(3 of 3), Hispanic Origin, Taken to Hospital, Related Factor(1)Person Level, Related Factor(2)Person Level, Related Factor(3)Person Level and Race. The dependent variable is the Injury Severity.
To investigate the influence of these factors on the seriousness of the injuries, the dependent factor is categorized into : No Injury, Fatal Injury, Possible Injury, Incapacitating Injury, Non Incapacitating Evident Injury, Injury Severity Unknown, Died Prior to Accident.

3.2 Data Preprocessing :

The Preprocessing step in building a deep learning model is the most essential part which takes upto 50-60 percent of the entire time taken to build the model. Data preprocessing is a technique of cleaning and organizing the raw data making it suitable for training and building a model.

In the dataset that is considered for this project there are around 26 categorical columns and each column consists of several categories. One Hot encoding is later performed on these categorical columns in order to convert them to numerical values. This process can be done easily if there are columns with few categories. But in case of the columns with many categories it becomes difficult.

Before the process of one hot encoding started, we first dropped those columns that were not so necessary for predicting the severity of the accident injuries. The columns to be dropped were chosen using Variance Threshold and by considering the Correlation between the features.

Variance Threshold : It is used to remove the low variant features and it can only be applied on the training data. In this project, we have checked for the constant columns by providing various threshold values from 0 to 0.5 and following are the constant columns for each threshold value provided:

Threshold 0 - ‘0 constant columns’

Threshold 0.1 - ‘0 constant columns’

Threshold 0.2 - ‘2 constant columns’

Threshold 0.3 - ‘4 constant columns’

Threshold 0.4 - ‘5 constant columns’

Threshold 0.5 - ‘6 constant columns’

```
In [38]: from sklearn.feature_selection import VarianceThreshold

In [39]: var_thres=VarianceThreshold(threshold=0.2)
         var_thres.fit(X_train)

Out[39]: VarianceThreshold(threshold=0.2)

In [40]: sum(var_thres.get_support())

Out[40]: 24

In [41]: constant_columns = [column for column in X_train.columns if column not in X_train.columns[var_thres.get_support()]]
         print(len(constant_columns))

         2

In [42]: for column in constant_columns:
             print(column)

          EXTRICATION
          METHOD_OF_DRUG_DETERMINATION

In [43]: X_train.drop(constant_columns, axis=1)
```

After checking the number of constant columns for each of the threshold values, the minimum threshold value for which the constant columns were not '0' is selected. In this case the threshold value 0.2 is selected for which we have 2 constant columns. Here, the two constant columns are dropped which are Extrication and Method Of Drug Determination.

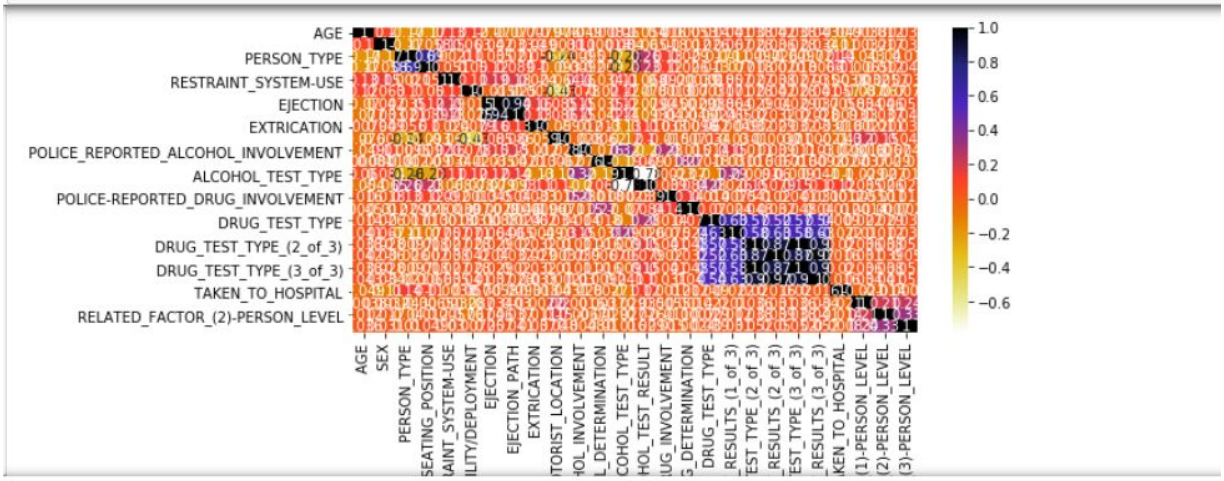
Correlation : After dropping the constant columns, the dataset is still not ready for training purpose. Therefore, we now used the concept of correlation in order to drop the unwanted features.

Threshold value determines the columns that are highly correlated to each other. Out of the columns that are said to be highly correlated, one of the columns is kept in the dataset and all the remaining columns are dropped.

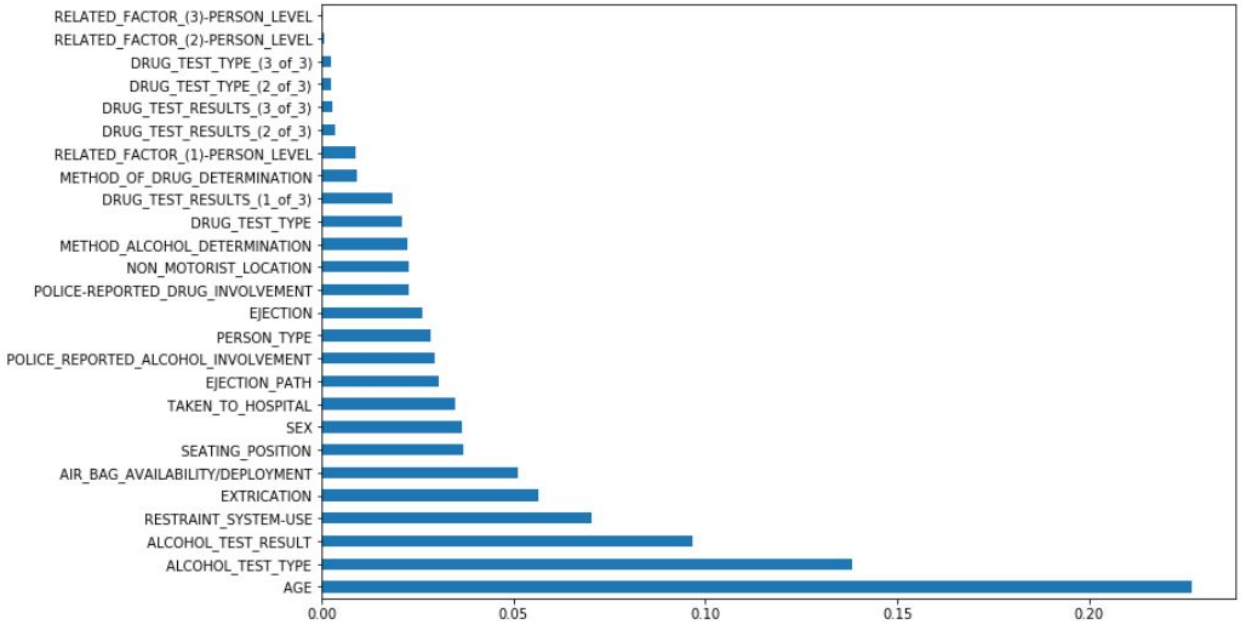
We have taken the threshold value as 90% (0.9) for which the highly correlated columns are 3. The following are the three highly correlated features which can be dropped :

- Drug test results (3 of 3)
- Drug test type (3 of 3)
- Ejection Path

```
import seaborn as sns
#pearson correlation
plt.figure(figsize=(9,4))
cor = X_train.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.CMRmap_r)
plt.show()
```



All of these 3 columns/features have been removed from the dataset and the process of building the model is continued. After dropping these 5 features from the dataset, we then manually dropped the features that have no effect on the prediction process which are the 'Case State', 'Hispanic Origin', 'Race'.



This graph shows the importance of the columns in regard to predicting the depth of the injuries caused by an accident. We then checked for the order of importance of the features present in the dataset in order to predict the severity of the accident injuries. Considering the graph that shows the order of importance of the features, six columns that are considered to have the least priority or the least involvement in the prediction process have been dropped which are ‘Method Of Drug Determination‘ , ‘Related Factor(3)Person Type, Related_Factor(2)-Person Level, Drug_Test_Type_(2_of_3), Drug_Test_Results_(2_of_3) and Method_Alcohol_Determination.

Now that we have a clean dataset with only necessary columns, we have performed the one hot encoding to turn the categorical data into numerical values.

4. LITERATURE SURVEY

Literature Survey is quite possibly the main strides to be thought of while building up a model. Before beginning the way toward building up the model, it is critical to comprehend what are the essential factors that are to be mulled over, how long will it require for the model to be created, are there any models or undertakings that are like the model that we are creating? If there are any such comparative activities it is imperative to see all the highlights present in the generally existing model and furthermore check with what are the additional highlights that you are adding to your model to separate it from the all around existing one. After this exploration is done, it is presently an opportunity to decide the tools that are needed for building up the model. Since all the data is gathered and handled we can begin constructing the project or the model according to the arrangement for the proposed system.

4.1 Existing System

The primary phase of road network security management is to distinguish unsafe street areas. The examination of literature shows that most of the nations utilize the system of utilization of dangerous street areas and an investigation of mishaps happening there. Nonetheless, none of these systems prevail with regards to coming nearer to anticipating the specific injury seriousness that is brought about by a mishap.

All the current models either are utilized for dealing with the traffic at the hour of the mishap or to simply see whether the individual who met with an accident is alive or dead or the models help in site management or in some cases they disclose to us how long has it been since the mishap occurred. There are no such models that disclose to us the seriousness levels of the injury.

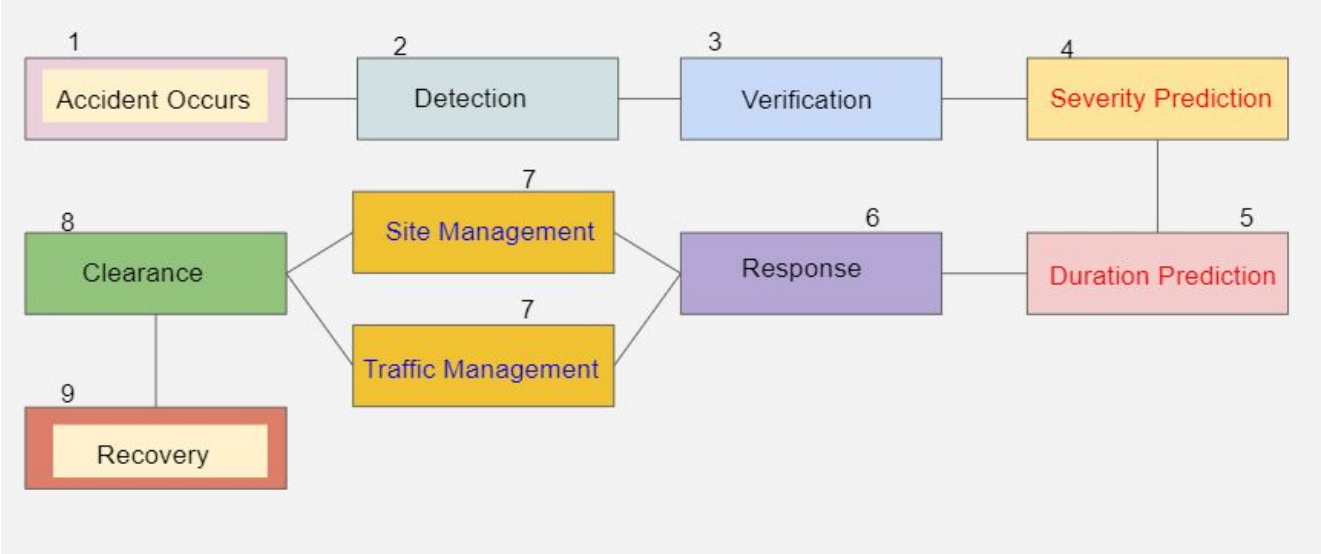


Fig:4.1.1 Existing Systems

4.2 Proposed System

In the proposed framework, Deep Neural Network is utilized to foresee the injury seriousness of the crashes dependent on 10k auto collision records. For every mishap record, 30 unique credits have been gathered at the hour of the mishap.

We will utilize the traffic accident information to construct classifiers in different ways. The entire dataset is used for training and testing, while 90% of the information is utilized for training and the excess 10% is utilized for testing. The outcomes in the wake of bunching uncovered huge improvement in the expectation exactness of the classifier.

For every mishap, the arranged model shows how likely this mishap would bring about each class.

- No Injury
- Fatal Injury
- Possible Injury
- Incapacitating Injury
- Non-Incapacitating Evident Injury
- Injury Severity Unknown
- Died Prior To Accident

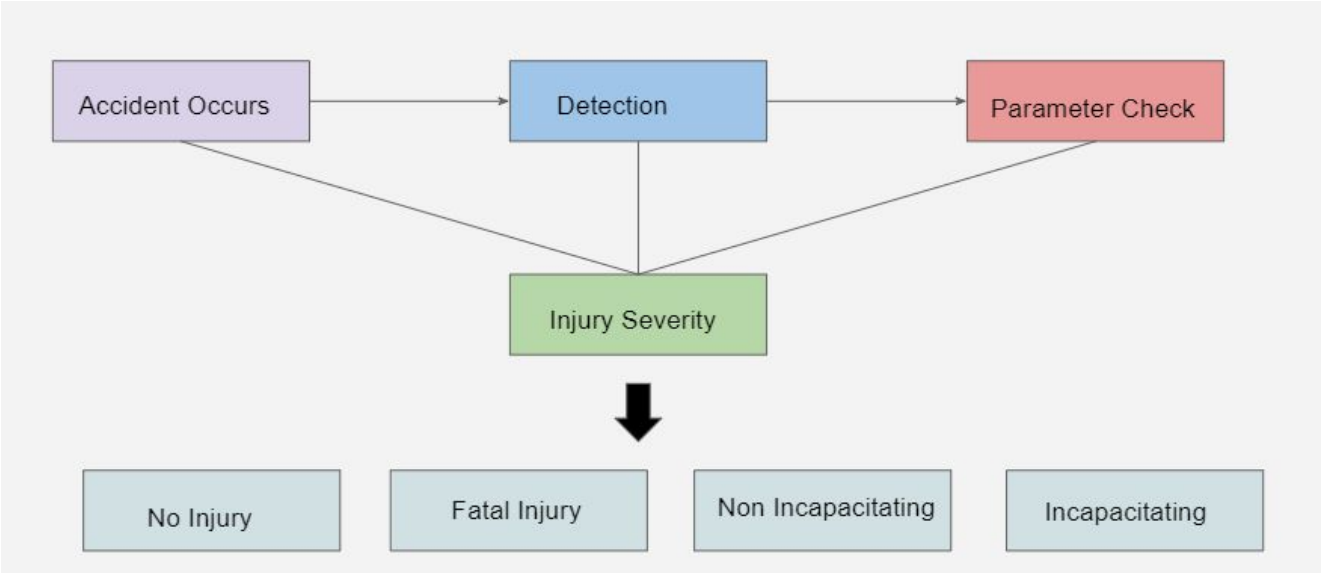


Fig:4.2.1 Proposed Mode

5. SCOPE AND SIGNIFICANCE

A very much characterized project extension is a need to guarantee the accomplishment of a task. Without it, regardless of how proficient, how compelling and how hard you work, you will not have the option to prevail in your project.

Characterizing the task scope involves embracing a clear vision and a concession to the results of the venture. This permits every achievement of the project to remain on track.

It additionally permits you to dole out assets and shape a sensible task course of events.

When we had this set up, we could assign tasks to ourselves all the more effectively and give guidance for every one of us in the group so that we can complete the project within a cutoff time.

5.1 Usecase /Application

There are numerous areas that are inclined to accidents and many individuals don't have the data about these zones. This is a primary explanation because of which the graph that shows the event of accidents at that spot probably won't have a declination in the bend.

We can gather the information of pretty much all the accidents that have occurred in an area and by utilizing the "Injury Severity Prediction Model", we can discover the earnestness of wounds that are brought about by the accidents that have happened in that specific spot. By doing this we can discover the absolute number of deaths that the area has a count of and furthermore the level of intense wounds that have happened.

By considering the total number of deaths and severe wounds, we by then can choose if that specific area is an accident inclined region or not.
We can also begin rating the regions as the most perilous accident inclined zones to the least accident inclined regions.

5.2 IMPORTANCE

The Accident Injury Severity Prediction model assumes a significant part by not just anticipating the injury seriousness but also in various perspectives.

A couple of it are, we can utilize this model close to the traffic lights where generally a lot of accidents occur. So by utilizing this model at these sorts of spots, when an accident is experienced, we can promptly check for the earnestness of the injury and make vital strides.

Likewise zebra intersections or commonly called as zebra crossings are the basic spots where the majority of the accidents occur and hence by utilizing this model there, we can make a quick move dependent on the profundity of the injury whether as to take the individual to the hospital or on the off chance that it is alright to simply give him/her medical aid or in the event that it is a crisis.

6. REQUIREMENTS

Requirements are something that is needed or that should be finished. The Project Management Body of Knowledge characterizes requirements as "a condition or capacity that is needed to be available in an item, administration, or result to fulfill an agreement or other officially forced determination."

Collecting prerequisites for an undertaking is an extremely crucial part. Indeed, gather prerequisites measure assists with characterizing project scope during scope management.

There are some arrangements of devices and procedures to assemble project necessities. It appears to be functional to gather all necessities toward the beginning utilizing a prerequisite which is the requirement-gathering tool.

Undertaking necessities are conditions or assignments that should be finished to guarantee the achievement or fulfillment of the venture. They give away work that should be finished. They're intended to adjust the task's assets to the goals of the association.

Here is a rundown of necessities that have been utilized in effectively assembling this model. They are :

- Windows 10 OS
- Python
- Matlab
- Tensorflow 2.3.0
- Keras 2.3.0
- Seaborn 0.9.0 to 0.10.1
- Scikit-learn 0.20.2 to 0.23.2


scikit
learn


Seaborn


Windows 10


NumPy


Pandas


K
Keras


MATLAB

6.1 Scikit-learn 0.20.2 to 0.23.2

Scikit-learn is a Python module for machine learning built on top of SciPy and is distributed under the 3-Clause BSD license.

Dependencies

Scikit-learn requires:

- Python (>= 3.6)
- NumPy (>= 1.13.3)
- SciPy (>= 0.19.1)
- joblib (>= 0.11)
- threadpoolctl (>= 2.0.0)

Scikit-learn is probably the most useful library for machine **learning** in Python. The **sklearn** library contains a lot of efficient tools for machine **learning** and statistical modeling including classification, regression, clustering and dimensionality reduction.

Scikit Learn is based on top of a few normal data and math Python libraries. Such a plan makes it excessively simple to coordinate between them all. You can pass numpy arrays and pandas data frames straightforwardly to the ML algorithms of Scikit! It utilizes the accompanying libraries:
NumPy: For any work with frameworks, particularly math tasks
SciPy: Scientific and specialized processing
Matplotlib: Data representation
IPython: Interactive support for Python
Sympy: Symbolic math
Pandas: Data taking care of, control, and examination

Scikit Learn is centered around Machine Learning, e.g data demonstrating. It isn't worried about the stacking, dealing with, controlling, and picturing of data. Consequently, it is normal and regular practice to utilize the above libraries, particularly NumPy, for those additional means; they are made for one another!

Scikit Learn also allows us to visualise our tree using the graphviz library. It comes with a few options that will help in visualising the decision nodes and splits that the model learned which is super useful for understanding how it all works.

Scikit's robust set of algorithm offerings includes:

- **Regression:** Fitting linear and non-linear models
- **Clustering:** Unsupervised classification
- **Decision Trees:** Tree induction and pruning for both classification and regression tasks
- **Neural Networks:** End-to-end training for both classification and regression. Layers can be easily defined in a tuple
- **SVMs:** for learning decision boundaries
- **Naive Bayes**: Direct probabilistic modelling

6.2 Seaborn

Seaborn is an astonishing Python representation library based on top of matplotlib.
It gives us the ability to make enhanced information visuals. This causes us to comprehend the information by showing it in a visual setting to uncover any concealed correlations between factors or patterns that probably won't be evident at first. Seaborn has an undeniable level interface when contrasted with the low degree of Matplotlib.

Seaborn makes our diagrams and plots look connected with and empowers a portion of the regular information perception needs (like planning tone to a variable or utilizing faceting). Essentially, it makes the information representation and investigation simple to overcome.

There are two or three (major) impediments in matplotlib that Seaborn fixes:
1. Seaborn accompanies countless significant level interfaces and tweaked subjects that matplotlib needs as it is difficult to sort out the settings that make plots appealing.
2. Matplotlib functions don't work admirably with data frames, though seaborn does.
The seaborn library has four required dependencies you need to have:
NumPy (>= 1.9.3)
SciPy (>= 0.14.0)
matplotlib (>= 1.4.3)
Pandas (>= 0.15.2)

6.3 Keras

Keras is a general library, utilized uniquely for building neural network models. It is written in Python and is viable with both Python – 2.7 and 3.5. Keras was explicitly created for quick execution of thoughts. It has a basic and profoundly measured interface, which makes it simpler to make even complex neural network models. This library abstracts low level libraries, to be specific Theano and TensorFlow so that the client is liberated from "implementation details" of these libraries.
The vital highlights of Keras are:
Modularity : Modules important for building a neural network are remembered for a basic interface so Keras is simpler to use for the end client.
Minimalistic : Implementation is short and succinct.
Extensibility : It's extremely simple to compose another module for Keras and makes it appropriate for advance exploration.

Keras permits clients to productize deep models on cell phones (iOS and Android), on the web, or on the Java Virtual Machine. It likewise permits utilization of dispersed preparing of deep-learning models on bunches of Graphics processing units (GPU) and tensor processing units (TPU).
The most compelling motivations to utilize Keras originate from its core values, fundamentally the one about being easy to use. Past simplicity of learning and simplicity of model structure, Keras offers the upsides of wide reception, uphold for a wide scope of production deployment alternatives, combination with at any rate five back-end engines (TensorFlow, CNTK, Theano, MXNet, and PlaidML), and solid help for various GPUs and appropriated preparing. In addition, Keras is sponsored by Google, Microsoft, Amazon, Apple, Nvidia, Uber, and others.

6.4 TensorFlow

"TensorFlow – The lingua franca of machine learning researchers and developers."
TensorFlow is an open source adaptable programming library for performing mathematical and graphical calculations utilizing information stream charts. An adaptable, versatile, and convenient framework utilized for making huge scope neural networks with various layers. The base language for TensorFlow is Python or C++. TensorFlow offers phenomenal compositional help that makes it simple to send complex mathematical calculations across assorted stages going from PC to mobiles, edge gadgets, and furthermore cluster of servers. TensorFlow has been intended for use in both research and development and in production systems.. TensorFlow may be a needless excess for easier errands however a solid wager for complex profound learning tasks.

"TensorFlow doesn't take care of the issue, yet gives you the tool compartment to extract away from academics of a convolutional neural net and utilize one to tackle our concern."

TensorFlow gives exceptionally adaptable and measured design meaning you can utilize just the necessary parts or utilize all the parts together.
TensorFlow coordinates with anything that can call a basic C API and furthermore manages restricted ideas, such as, sessions, Tensors and a DAG. Computations should be communicated as an information stream diagram and TensorFlow gives various renditions of similar models or numerous models for execution. Engineers can part the plan of the information stream from its execution. Develop the data flow diagram and afterwards send it for execution on the CPUs of machines or to the GPU's or a mix of the two. This takes place through a solitary interface concealing all the intricacies from the client. Since the execution is offbeat it scales across numerous machines and can handle huge volumes of information. This encourages non-programmed movement to new models/renditions and A/B testing of exploratory models."

6.5 Matlab

MATLAB is a superior language for specialized processing. It incorporates computation, visualization, and programming in a simple to-utilize way where issues and solutions are communicated in natural numerical documentation. Commonplace uses include:

Math and calculation

Calculation advancement

Demonstrating, reenactment, and prototyping

Data analysis, exploration, and visualization

Logical and designing illustrations

Application advancement, including Graphical User Interface building

MATLAB is an intelligent framework whose fundamental information component is a cluster that doesn't need dimensioning. This permits you to take care of numerous specialized registering issues, particularly those with matrix and vector definitions, in a small portion of

the time it would take to compose a program in a scalar noninteractive language, for example, C or Fortran.

The name MATLAB represents a grid research center. MATLAB was initially composed to give simple admittance to framework programming created by the LINPACK and EISPACK projects, which together address the cutting edge in programming for lattice calculation.

MATLAB has advanced over a time of years with contributions from numerous clients. In university conditions, it is the standard instructional device for initial and progressed courses in arithmetic, designing, and science. In industry, MATLAB is the instrument of decision for high-efficiency examination, improvement, and investigation.

MATLAB highlights a group of utilization explicit arrangements called tool stash. Vital to most clients of MATLAB, tool kits permit you to learn and apply particular innovation. Toolboxes are exhaustive assortments of MATLAB capacities (M-documents) that stretch out the MATLAB climate to take care of specific classes of issues. Regions in which tool stash are accessible incorporate sign handling, control frameworks, neural organizations, fluffy rationale, wavelets, recreation, and numerous others.

7. SYSTEM DESIGN

System design is the phase that bridges the gap between problem domain and the existing system in a manageable way. This phase focuses on the solution domain, i.e. “how to implement?”

It is the stage where the SRS record is changed over into a format that can be actualized and chooses how the system will work.

In this stage, the intricate movement of system improvement is isolated into a few more modest sub-exercises, which organize with one another to accomplish the fundamental target of system advancement.

7.1 Architecture Diagram

Numerous individuals confound the two, Design and architecture diagram. Yet, they're totally various things. An architectural outline portrays what you're building, and where the limitations lie. A design outline discloses how to construct it. A picture is worth a thousand words, or so the truism goes. Likewise, architectural diagrams help pass on complex data in a solitary picture.

Architectural diagrams show frameworks. Showing data outwardly allows us to see everything at a glance, including how things communicate. This is particularly helpful when making changes: We will be able to see the downstream impacts of a given change all the more obviously.

Architectural diagrams additionally separate complex frameworks and cycles into layers. So instead of attempting to understand everything simultaneously, we can zoom in and center around more modest sub-cycles or frameworks.

One of the principle issues computer programmers face is consistency. At the point when we are chipping away at anything that includes various individuals, there's consistently a danger of miscommunication and disparities between project groups and designers. It is essential to normalize data, which is the place where an architectural chart proves to be useful.

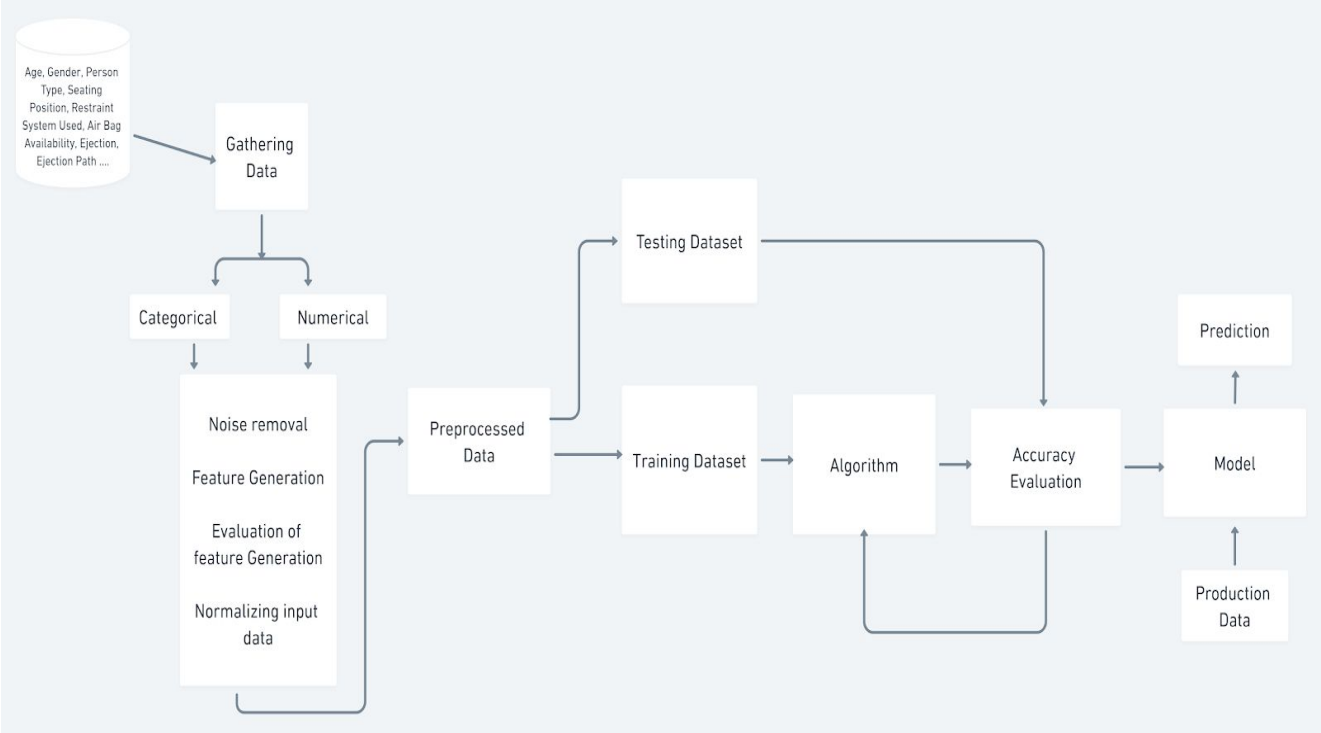


Fig:7.1.1 Architecture Diagram

7.2 UML Diagrams

Unified Modeling Language is a detailed language that is utilized in the programming field. It can be characterized as a broadly useful language that utilizes a graphical assignment which can make an abstract model. This abstract model would then be able to be utilized in a framework. This framework is known as the UML model. The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the

artifacts of software systems, just as for business demonstrating and other non-programming frameworks. This investigation centers around the significance of UML charts (class, use case, sequence, activity, component, deployment, collaboration) for Large-sized tasks for example s/w that expects 1 to 2 years for improvement. It has been discovered that all the UML graphs are not as often as possible utilized during Large-sized s/w improvement. According to the information investigation Class diagram considered as the main outline among all UML charts while Collaboration diagram considered as least significant outline. Sequence diagram has a slight advantage over the use case graph and treats hardly more significant charts. Likewise Deployment diagram assumes imperative part than component diagram during s/w advancement.

7.2.1 Use Case Diagram

Use case diagrams are used to assemble the prerequisites of a system including internal and external impacts. These prerequisites are generally plan necessities. So when a system is examined to assemble its functionalities use cases are prepared and actors are recognized. The reason for use case graphs is to catch the unique part of a framework. Presently when the underlying task is finished use case charts are displayed to introduce the external view.
So in a nutshell, the reasons for use case graphs can be as follows:

- Used to accumulate prerequisites of a system.
- Used to get an external perspective on a system.
- Identify outer and inward factors affecting the framework.
- Show the associating among the prerequisites are actors.

Use case graphs are considered for undeniable level necessity investigation of a framework. So when the requirements of a system are analyzed the functionalities are captured in use cases.

So we can say that use cases are only the framework functionalities written in a coordinated way. Presently the second things which are applicable to the use cases are the actors. Actors can be characterized as something that connects with the framework.

The actor can be a human user, some inner applications or might be some outer applications. So in short when we are intending to draw a use case outline we ought to have the accompanying things distinguished.

•Functionalities to be addressed as a use case
•Actors
•Relationships among the use cases and actors.

Use case graphs are attracted to catch the useful prerequisites of a system. So subsequent to distinguishing the above things we need to follow the accompanying rules to draw a productive use case graph.
•The name of a use case is vital. So the name ought to be picked in a way with the goal that it can recognize the functionalities performed.
•Give a reasonable name for actors.
•Show connections and conditions clearly in the diagram.
•Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.
•Use notes at whatever point needed to explain some significant focuses.

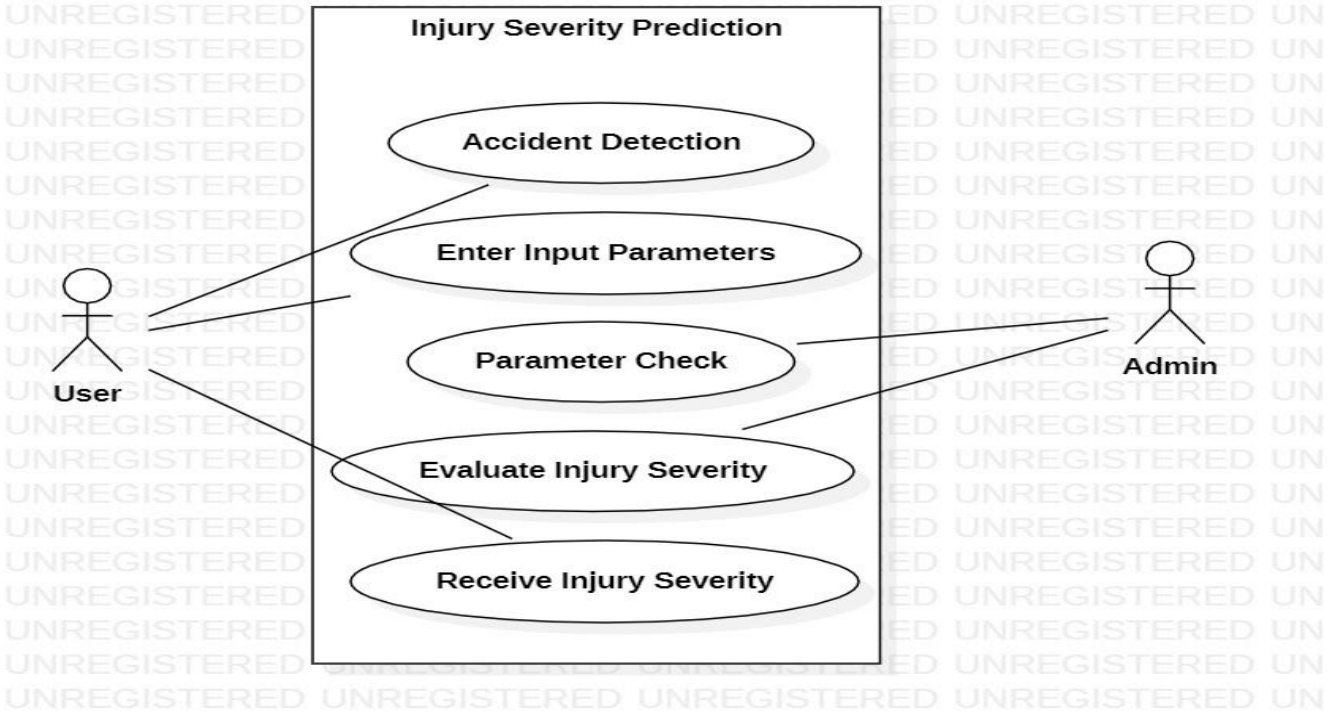


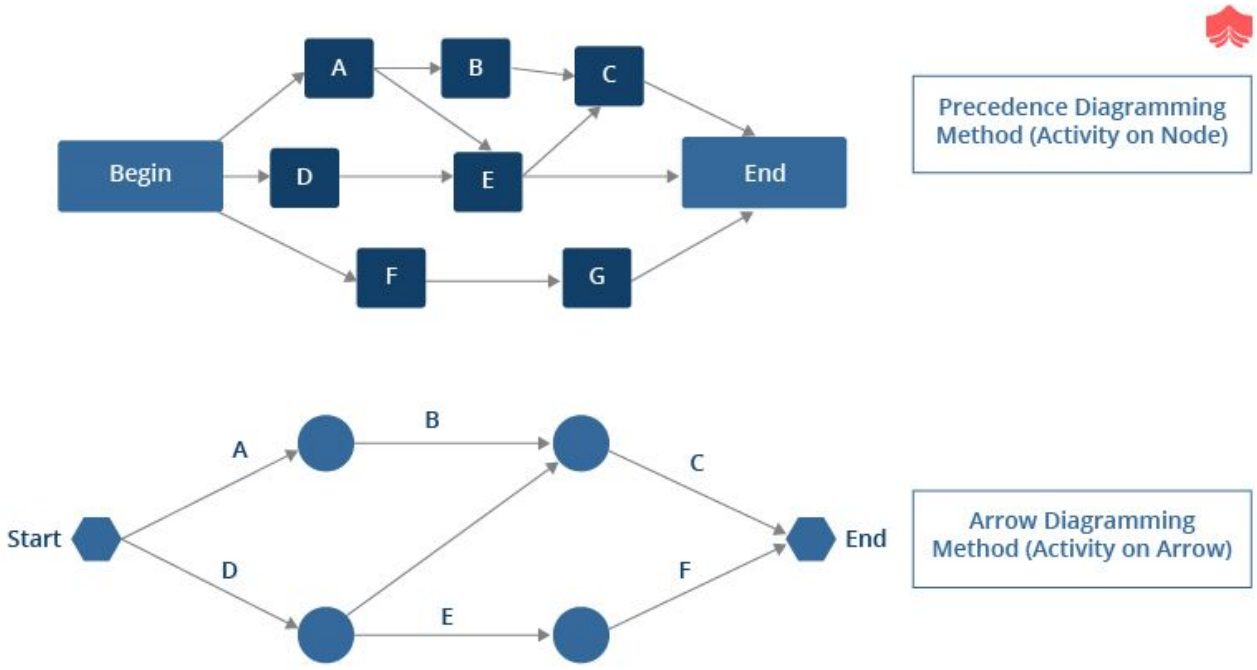
Fig 7.2.1(a) Use Case Diagram

7.2.2 Activity Diagram

A project will comprise a number of tasks to be performed. These tasks should be performed by considering all intelligence conditions. A portion of the tasks must be acted in pure sequence while numerous tasks should be possible simultaneously.

We will inspect the rundown of tasks and will choose the conditions between them to think of a request for sequence where they will be completed. Each undertaking will have an archetype and furthermore a replacement. Making a correct sequence of tasks will guarantee that the tasks are done appropriately and the last expectations come out pleasantly. Ideal sequence will likewise guarantee that we can finish all the tasks in the most ideal time by utilizing simultaneousness.

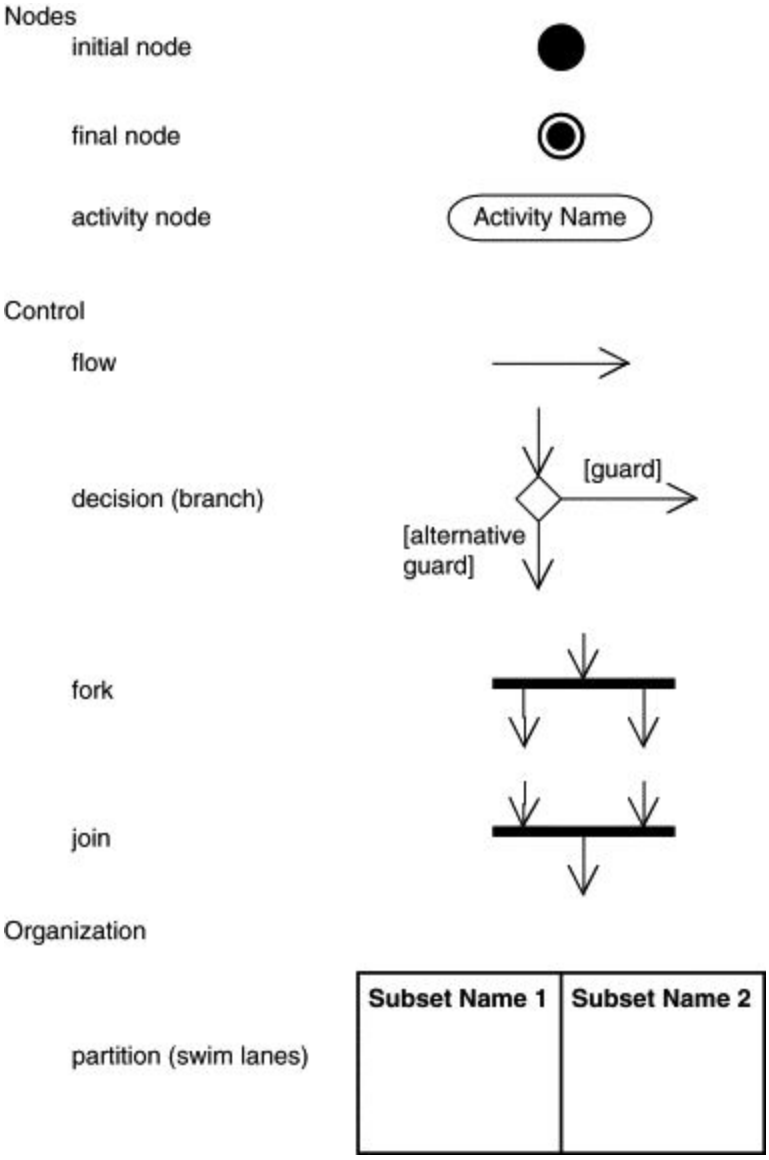
There are 2 unique methods for making a movement sequence graph as underneath:
Precedence Diagramming Method (PDM) – This is otherwise called Activity on Node (AON)
Arrow Diagramming Method (ADM) – This is otherwise called Activity on Arrow (AOA)



Activity diagrams show the progression of control and activities as rounded rectangles. Activities are commonplace activity states – states that travel consequently to the following state after the activity is finished. The filled-in circle addresses the beginning of the activity outline where the progression of control begins. Arrows address the change starting with one activity then onto the next. Synchronization bars show how activities occur in equal amounts and watch the change.

UML is flexible in nature; so activity diagrams might be utilized toward the start of the existence cycle or in various stages totally. Numerous individuals use activity diagrams to show the stream between the techniques for a class. They can be utilized as swim paths which plainly show to all gatherings the individual or gathering liable for a particular activity.

"Activity Diagrams for Workflow" section.



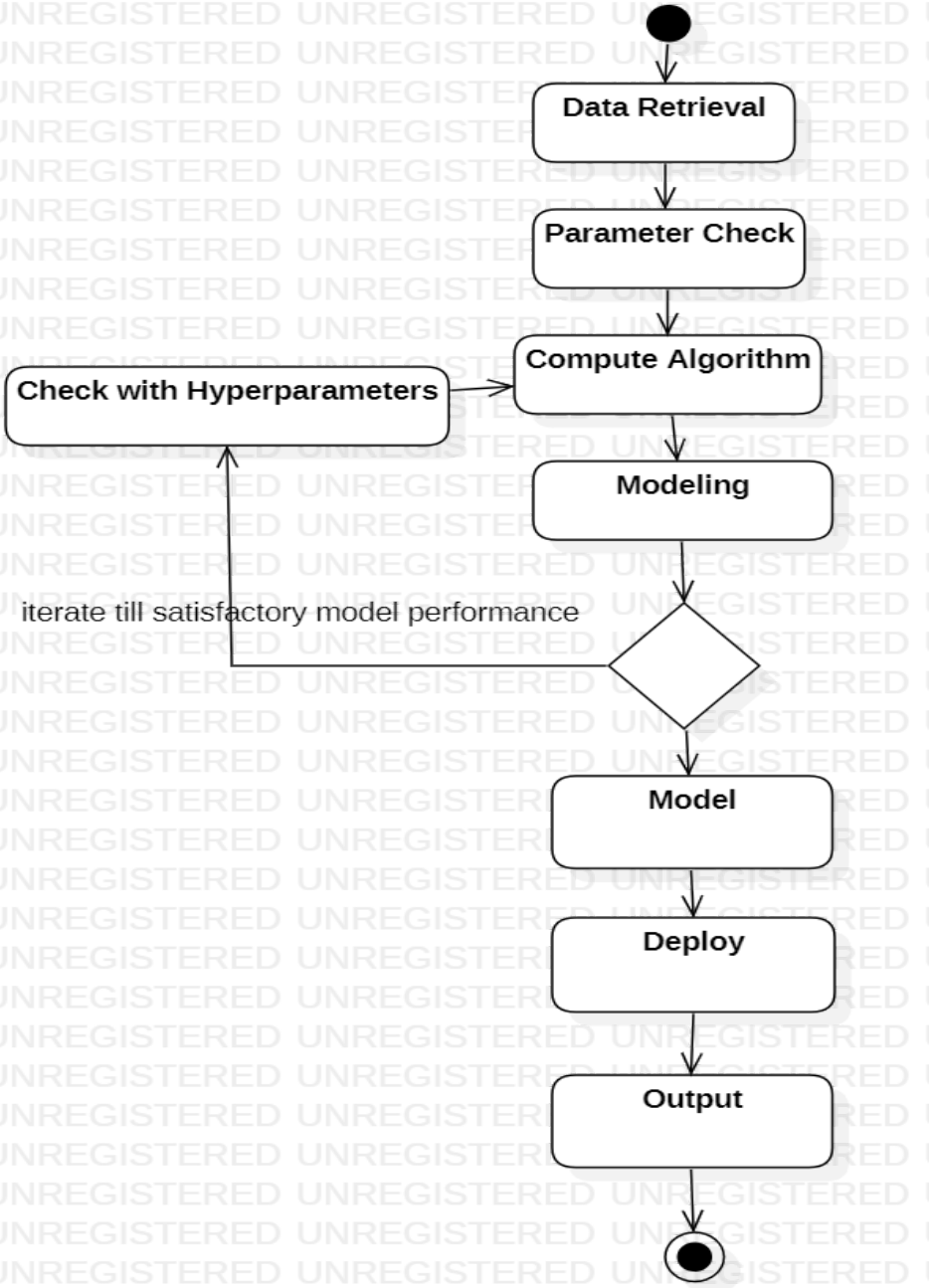


Fig 7.2.2(A) Activity Diagram

8. IMPLEMENTATION

During the execution stage, the venture plan is kicked off and work of the project is performed. It is essential to keep up control and communicate depending on the situation during usage. Progress is consistently observed and suitable changes are made and recorded as fluctuations from the first arrangement. In any task, more often than not is spent in this progression. During project usage, individuals are doing the assignments, and progress data is being accounted for through customary group gatherings. The group utilizes this data to keep up power over the heading of the venture by contrasting the advancement reports and the undertaking intends to quantify the presentation of the task exercises and make a remedial move depending on the situation. The primary strategy ought to consistently be to bring the undertaking back on course (i.e., to restore it to the first arrangement). On the off chance that that can't occur, the group should record varieties from the first arrangement and record and distribute alterations to the arrangement.

Status reports ought to consistently underline the foreseen end point as far as cost, timetable, and nature of expectations. Each task deliverable created ought to be looked into for quality and estimated against the acknowledgment measures. When the entirety of the expectations have been created and the client has acknowledged the last arrangement, the venture is prepared for conclusion.

8.1 Neural Networks

The human brain, its capacities, and the manner in which it works filled in as the motivation for the making of the neural network. Artificial intelligence and machine learning, which is a subset of AI, have a fundamental influence in its usefulness. It begins working when an engineer enters information and assembles a machine learning calculation, generally utilizing the "if ... else ..." rule of building a program. The deep neural network doesn't just work as per the calculation yet additionally can foresee an answer for an undertaking and make

conclusions utilizing its past experience. For this situation, you don't have to utilize programming or coding to find a solution.

8.1.1 Artificial Neural Network :

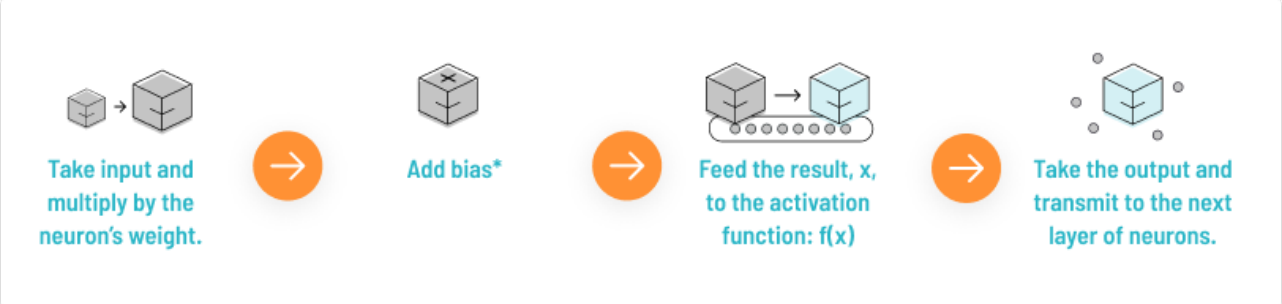
Initially enlivened by neurobiology, deep neural network models have gotten a useful asset of AI and man-made consciousness. They can surmise functions and elements by learning from examples.

First presented during the 1960s, the ANN can take care of numerous complex logical issues. It is a biologically-motivated machine learning tool that captures and represents extremely complex non-linear relationships existing in real work data sets.

It works by mirroring the neurological functions of the human cerebrum, similarly as neurons invigorate and respond to a circumstance in the human mind. It can anticipate the result of a perception dependent on the example obtained from authentic information in the wake of conveying a preparation technique.

An artificial neuron is the fundamental unit of a neural network. Input nodes move the information to a neuron, which is prepared inside to deliver a reaction. ANN is handled in two stages: the initial step linear combination of input values and then the obtained results are used as an argument for non-linear activation function. Every association has a weight allocated to it and the activation function is differentiable.

The basic process carried out by a neuron in a neural network is:



Network engineering is characterized by neuronal association. The engineering comprises an input layer, a hidden layer, and an output layer. The input, hidden and output layers comprise of five, three, and one neuron, separately. The output of one layer is used as input to the following layer. The activation function in the neuron combines the inputs by multiplying with the corresponding weights.

There is additionally an inclination segment in every neuron. Enhancement techniques are utilized to appraise the loads of the information ("called the preparation of network") by limiting the loss function. A few preparing calculations are accessible and one of them is backpropagation, which depends on an inclination drop strategy for boundary assessment .

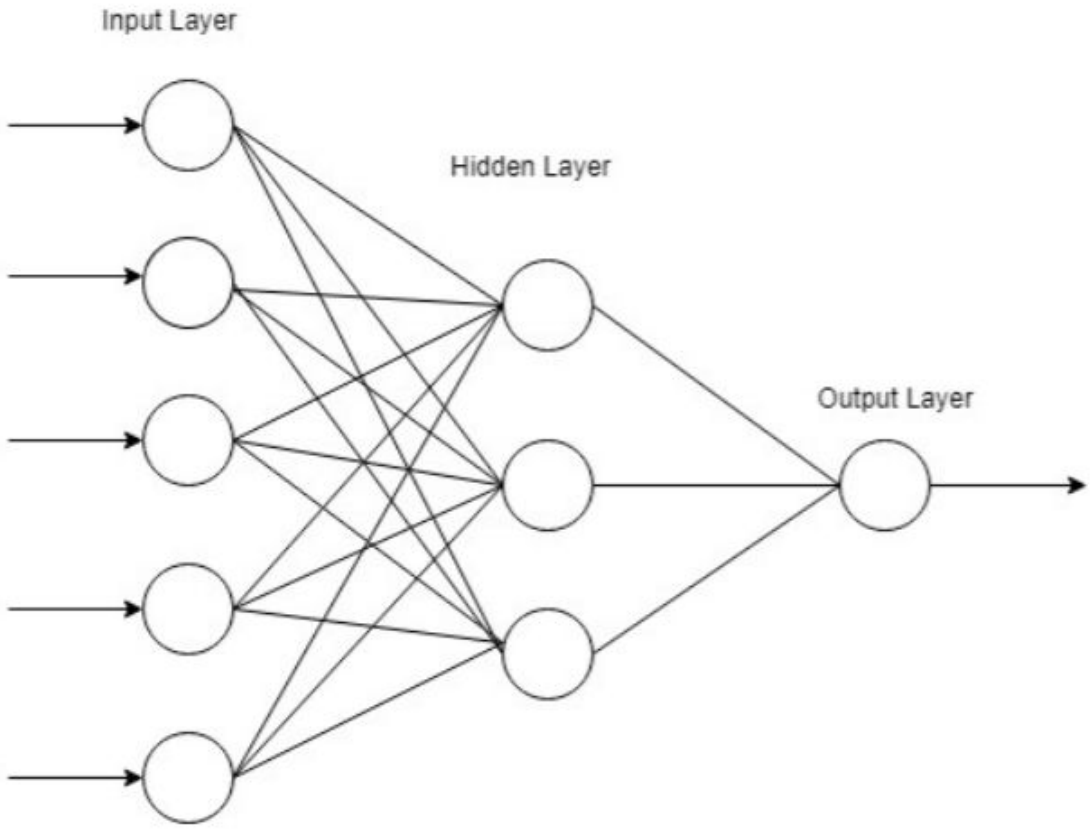


Figure 1. Typical feed-forward neural network.

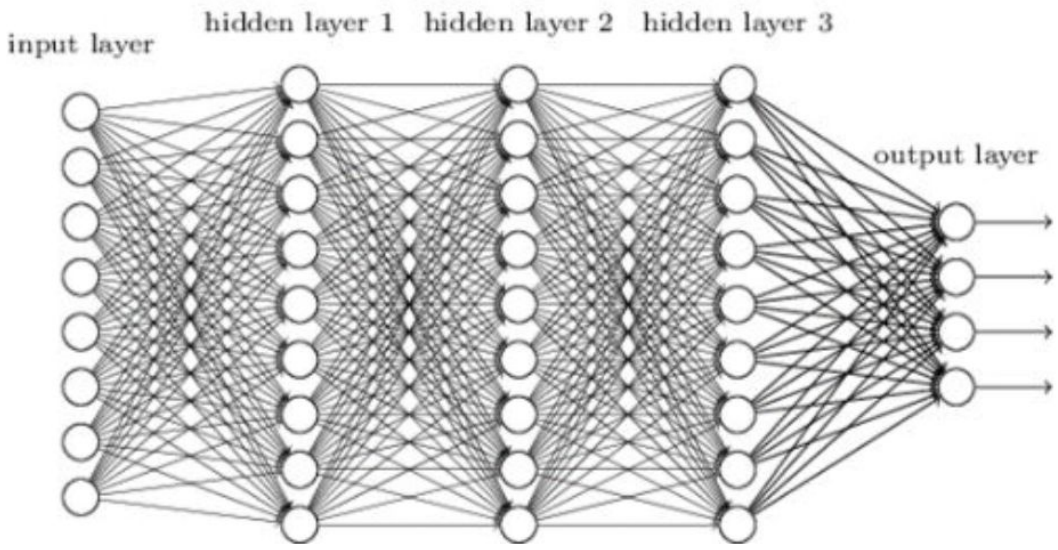
8.1.2 Deep Neural Networks

A deep neural network (DNN) is an artificial neural network (ANN) with different layers between the input and output layers. There are various sorts of neural networks yet they generally comprise similar parts: neurons, synapses, weights, biases, and functions. These parts work like the human brains and can be prepared like any other ML algorithm.

DNNs can show complex non-straight connections. DNN designs create compositional models where the item is communicated as a layered piece of natives. The additional layers empower organization of highlights from lower layers, possibly demonstrating complex information with less units than a correspondingly performing shallow organization. For example, it was demonstrated that scanty multivariate polynomials are dramatically simpler to rough with DNNs than with shallow organizations.

In a Deep Neural Network nodes are little pieces of the framework, and they resemble neurons of the human cerebrum. At the point when an improvement hits them, an interaction happens in these nodes. Some of them are associated and checked, and some are not, however all in all, nodes are assembled into layers.

The framework should deal with layers of information between the input and output to address an errand. The more layers it needs to cycle to get the outcome, the deeper the network is thought of. There is an idea of Credit Assignment Path (CAP) which implies the quantity of such layers required for the framework to finish the undertaking. The neural network is deep if the CAP list is more than two.



8.2 Steps Involved In Building The Neural Network

The first step towards building the neural network is to import tensorflow and keras. TensorFlow is an open source programming library for mathematical calculation utilizing data flow diagrams. Nodes in the diagram address numerical tasks, while graph edges address multi-dimensional data arrays (otherwise known as tensors) conveyed between them. The flexible architecture permits us to send calculations to at least one CPUs or GPUs in a desktop, server, or mobile device with a single API.

Keras is an incredible and simple to-utilize free open source Python library for creating and assessing deep learning models. It wraps the proficient mathematical calculation libraries Theano and TensorFlow and permits us to characterize and prepare neural network models in only a couple lines of code.

Alongside keras and tensorflow, we also import sequential from keras.model and dense, activation from keras.layers. A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor. Dense layer is the regular deeply connected neural network layer. It is the most common and frequently used layer. In artificial neural networks, the activation function of a node defines the output of that node given an input or set

of inputs. This is similar to the behavior of the linear perceptron in neural networks.

Now that the tensorflow and keras are imported we proceed to the next step which is to try to import the data from keras and then load it into a variable. Now this data is split into training and testing datasets. In the next step the scaling of the data is performed.

We now import SGD which is called the Stochastic Gradient Descent from tensorflow.keras.optimizers. Stochastic gradient descent is an extremely mainstream and common algorithm utilized in different Machine Learning calculations, above all structures the premise of Neural Networks.

Gradient, in plain terms implies slope or inclination of a surface. So gradient descent in a real sense implies descending a slope to arrive at the absolute bottom on that surface.

"Gradient descent is an iterative calculation, that begins from an irregular point on a capacity and goes down its slope in strides until it arrives at the absolute bottom of that function."

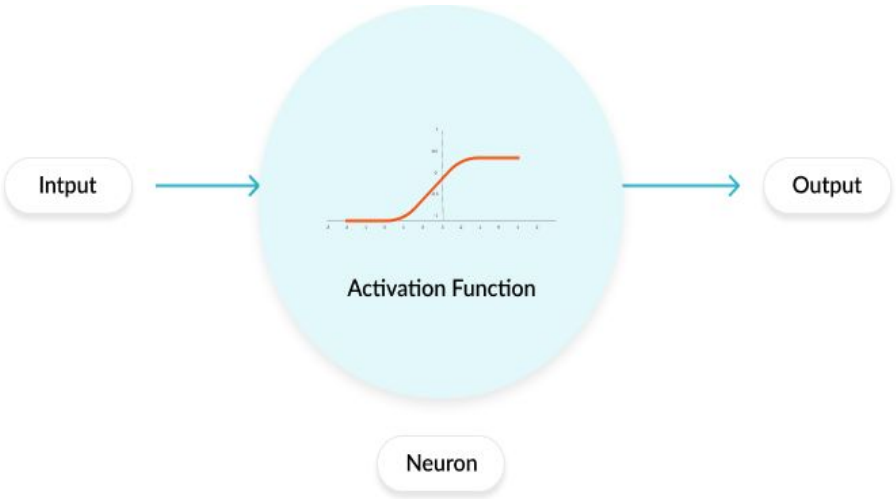
This algorithm is useful in situations where the optimal points can't be found by equating the slope of the capacity to 0. On account of direct relapse, we can intellectually plan the amount of squared residuals as the capacity "y" and the weight vector as "x" .

```
opti = SGD(lr=0.01, momentum=0.9)
```

```
modell = Sequential()
modell.add(Dense(600,activation='relu',input_dim=X_test.shape[1]))
modell.add(Dense(400,activation='relu'))
modell.add(Dense(200,activation='relu'))
modell.add(Dense(100,activation='relu'))
modell.add(Dense(50,activation='relu'))
modell.add(Dense(20,activation='relu'))
modell.add(Dense(8,activation='softmax'))
#model.add(Dense(1,activation='relu'))
modell.compile(optimizer=opti,loss='sparse_categorical_crossentropy',metrics=['accuracy'])
his=modell.fit(X_train,Y_train,batch_size=32,epochs=100,verbose=1)
```

```
Epoch 1/100
2840/2840 [==============================] - 179s 63ms/step - loss: 0.9054 - accuracy: 0.6307s - loss: 0.9057 - ac
Epoch 2/100
2840/2840 [==============================] - 91s 32ms/step - loss: 0.8457 - accuracy: 0.6493
Epoch 3/100
2840/2840 [==============================] - 90s 32ms/step - loss: 0.8318 - accuracy: 0.6546
Epoch 4/100
2840/2840 [==============================] - 112s 39ms/step - loss: 0.8238 - accuracy: 0.6581
Epoch 5/100
2840/2840 [==============================] - 99s 35ms/step - loss: 0.8158 - accuracy: 0.6607
Epoch 6/100
2840/2840 [==============================] - 105s 37ms/step - loss: 0.8099 - accuracy: 0.6611
Epoch 7/100
2840/2840 [==============================] - 124s 44ms/step - loss: 0.8038 - accuracy: 0.6639
Epoch 8/100
2840/2840 [==============================] - 104s 37ms/step - loss: 0.7997 - accuracy: 0.6651
Epoch 9/100
2840/2840 [==============================] - 114s 40ms/step - loss: 0.7939 - accuracy: 0.6676s - loss: 0.7939 - accura
Epoch 10/100
```

8.2.1 Activation Function



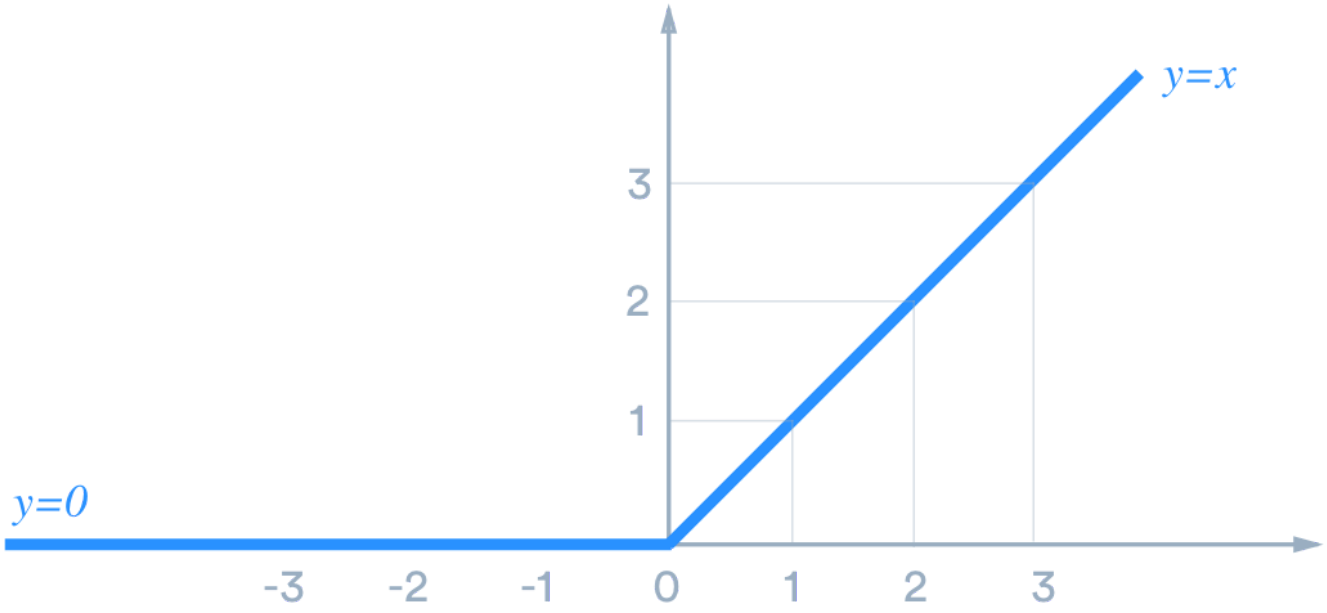
Activation functions are numerical conditions that decide the yield of a neural network. The function is connected to every neuron in the network, and decides if it ought to be enacted ("terminated") or not, founded on whether every neuron's info is pertinent for the model's expectation. Activation functions additionally help normalize the yield of every neuron to a reach somewhere in the range of 1 and 0 or between - 1 and 1.

An extra part of activation functions is that they should be computationally productive on the grounds that they are determined across thousands or even a large number of neurons for every information test. Present day neural networks utilize a strategy called backpropagation to prepare the model, which puts an expanded computational strain on the activation function, and its subsidiary function.

In this case we have model = sequential() and all the layers have been mentioned within the sequential function which means that all these layers are sequential layers i.e., from the first layer to the last layer and they are all connected to each other through some hidden layers and contain one input layer and one output layer.The activation function is taken as ‘relu’.

8.2.1(a) RELU (Rectified Linear Unit)

The rectified linear activation function or ReLU for short is a piecewise linear capacity that will yield the information straightforwardly on the off chance that it is positive, else, it will yield zero. ReLU represents a rectified linear unit, and is a sort of activation function. ReLU is the most commonly used activation function in neural networks.
Numerically, it is characterized as y = max(0, x).



As you can see, the ReLU is half rectified (from bottom). f(x) is zero when x is less than zero and f(x) is equal to x when x is above or equal to zero.

8.2.1(b) Softmax

Softmax is a numerical function that changes over a vector of numbers into a vector of probabilities, where the probabilities of each worth are corresponding to the overall size of each incentive in the vector.

The most well-known utilization of the softmax function in applied AI is in its utilization as an activation function in a neural network model. In particular, the network is arranged to yield N esteems, one for each class in the classification task, and the softmax function is utilized to standardize the outputs, changing over them from weighted total qualities into probabilities that total to one. Each value in the output of the softmax function is interpreted as the probability of membership for each class.

The softmax function is utilized as the activation function in the yield layer of neural network models that foresee a multinomial probability distribution.

That is, softmax is utilized as the activation function for multi-class classification issues where class enrollment is needed in excess of two class names.

The function can be utilized as an activation function for a hidden layer in a neural network, albeit this is more uncommon. It could be utilized when the model inside requirements to pick or weigh numerous various contributions at a bottleneck or connection layer.

In the Keras deep learning library with a eight-class classification task, use of softmax in the output layer may look as follows:

```
modell.add(Dense(8,activation='softmax'))
```

8.2.2 Loss Function

The purpose of loss functions is to compute the quantity that a model should seek to minimize during training.

Categorical Crossentropy

Categorical crossentropy is a loss function that is utilized in multi-class classification assignments. These are undertakings where a model can just have a place with one out of numerous potential classes, and the model should choose which one. Officially, it is intended to measure the distinction between two probability dispersions.

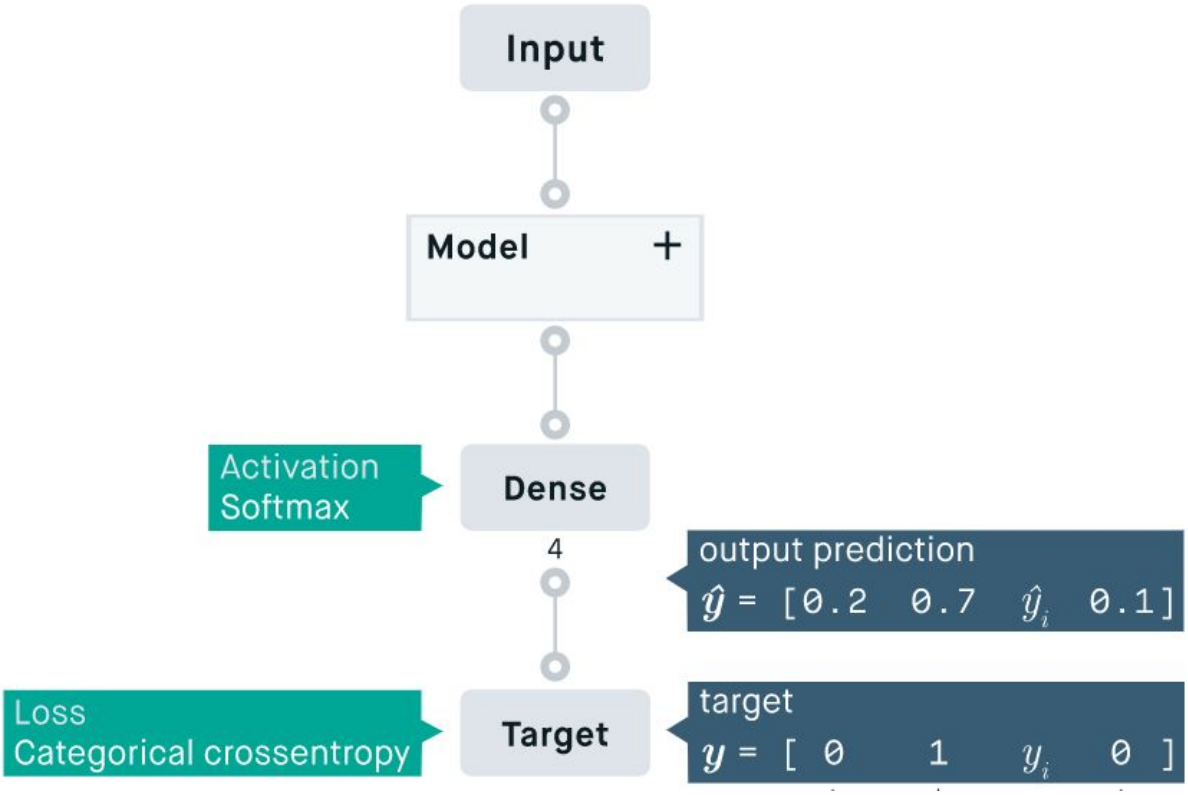


Fig 8.2.2(a) Categorical Crossentropy

We use this crossentropy loss function when there are two or more label classes. In this case we provide labels as integers. If we want to provide labels using one-hot representation, we use CategoricalCrossentropy loss.

As one of the multi-class, single-label classification datasets, the task is to classify the injury severity caused by an accident into eight categories. Let's build a Keras model to handle it with the last layer applied with "softmax" activation which outputs an array of eight probability scores. Each score will be the probability that belongs to one of our eight classes.
For a model with yield state of (None, 8), the traditional path is to have the target outputs changed over to the one-hot encoded exhibit to coordinate with the yield shape, nonetheless,

with the assistance of the sparse_categorical_crossentropy loss function, we can avoid that progression and keep the numbers as targets.

All you require is supplanting categorical_crossentropy with sparse_categorical_crossentropy when assembling the model this way.

```
modell.compile(optimizer=opti,loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

Sparse_Categorical_Crossentropy

Preparing a neural network includes passing information forward, through the model, and contrasting predictions and ground truth labels. This correlation is done by a loss function. In multiclass grouping issues, categorical crossentropy loss is the loss function of decision. Notwithstanding, it necessitates that our labels are one-hot encoded, which isn't generally the situation.

Therefore, in that case, sparse categorical crossentropy loss can be a decent decision. This loss function plays out a similar kind of loss – categorical crossentropy loss does – however works on integer targets rather than one-hot encoded ones.

8.2.3(a) Save the Model

We can save our trained model into a file using the Keras Library. It can be done as follows:

model.save(“name of the file in which we want to save the model”)

Ex : filename = “model.h5”

model.save(filename)

Here .h5 is the extension we use to save a HDF5 file.

```
filename = "model.h5"
model.save(filename)
```

8.2.3(b) Loading the saved model into the Machine Learning Program

The saved model can again be loaded into the machine learning program by importing load_model from tensorflow.keras.model

This loaded model can now be stored in a variable and can be used during the prediction process.

```
from keras.models import load_model
```

```
mod = load_model('model71.h5')
# summarize model.
#mod.summary()
```

8.2.4 PREDICTION

Step 1 : Import all the essential packages that are required.

Step 2 : Read the test dataset for which the predictions are to be finished.

Step 3 : Normalize the data(Scale the information)

Step 4 : Load the prepared model that is saved, utilizing load_model

Step 5 : Make Predictions on the test dataset.

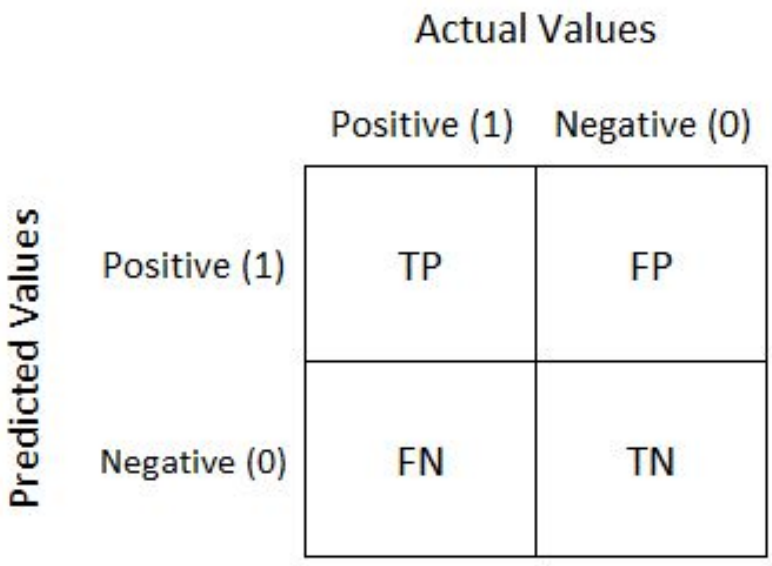
Step 6 : The predicted values are now compared with the actual dataset using a confusion matrix.

Making the predictions

```
prediction=pd.DataFrame(y_predicted_labels,columns=["Injury_severity(prediction)"])
prediction=prediction.replace({0:'Died_prior_to_accident',1:'Fatal_Injury',2:'Incapaciting_Injury',3:'Injured_Severity_Unknown',
```

Confusion Matrix

A vastly improved approach to evaluate the performance of a classifier is to take a gander at the confusion matrix. The overall thought is to check the number of times instances of class A are delegated as class B. For instance, to know the number of times the classifier confounded pictures of 5s with 3s, we would glance in the fifth row and third column of the confusion matrix. Each row in a confusion matrix addresses a real class, while every column addresses an anticipated class.



True Positive - If you test a condition which is true and the output additionally is predicted as true then it is called as True Positive.

True Negative - If you test for a condition which is false and the prediction is additionally false then it is True Negative.

False Positive - If you test for a condition which is false and the prediction gives the incentive as true then it is False Positive.

False Negative - If you test for a condition which is true and the predicted esteem is false then it is False Negative.

9. RESULT / OUTPUT

making the predictions

```
In [7]: prediction=pd.DataFrame(y_predicted_labels,columns=["Injury_severity(prediction)"])
        prediction=prediction.replace({0:'Died_prior_to_accident',1:'Fatal_Injury',2:'Incapaciting_Injury',3:'Injured_Severity_Unknown',4

In [8]: prediction
```

Out[8]:

	Injury_severity(prediction)
0	Fatal_Injury
1	Fatal_Injury
2	Nonincapaciting_Evident_Injury
3	Nonincapaciting_Evident_Injury
4	No_Injury
5	Fatal_Injury
6	Fatal_Injury
7	Incapaciting_Injury
8	Fatal_Injury
9	No_Injury
10	Nonincapaciting_Evident_Injury
11	Nonincapaciting_Evident_Injury

Fig 9.1 Prediction on Test Dataset

Confusion metrics

```
In [19]: from sklearn import metrics
         from sklearn.metrics import confusion_matrix

In [20]: cm=confusion_matrix(prediction,test)
         sns.heatmap(cm,annot=True)
```

Out[20]: <AxesSubplot:>

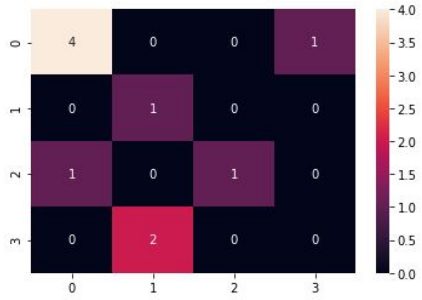


Fig 9.2 Confusion Matrix

Fig 9.2 shows the comparison of the predicted values with the values in the actual dataset using a confusion matrix.

```
In [21]: print(metrics.confusion_matrix(test,prediction, labels=['Died_prior_to_accident','Fatal_Injury','Incapaciting_Injury','Injured_Se
         # Printing the precision and recall, among other metrics
         print(metrics.classification_report(test,prediction, labels=['Died_prior_to_accident','Fatal_Injury','Incapaciting_Injury','Injur
```

```
[[0 0 0 0 0 0 0 0]
 [0 4 0 0 1 0 0 0]
 [0 0 1 0 0 2 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0]
 [0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]]
                               precision    recall  f1-score   support

       Died_prior_to_accident       0.00      0.00      0.00         0
                 Fatal_Injury       0.80      0.80      0.80         5
            Incapaciting_Injury       1.00      0.33      0.50         3
      Injured_Severity_Unknown       0.00      0.00      0.00         0
                     No_Injury       0.50      1.00      0.67         1
Nonincapaciting_Evident_Injury       0.00      0.00      0.00         1
               Possible_Injury       0.00      0.00      0.00         0
                       Unknown       0.00      0.00      0.00         0

                     micro avg       0.60      0.60      0.60        10
                     macro avg       0.29      0.27      0.25        10
                  weighted avg       0.75      0.60      0.62        10
```


Fig 9.2.1 Confusion Matrix

10. FRONTEND

Front-end web improvement is the act of changing information over to a graphical interface, using HTML, CSS, and JavaScript, so clients can see and communicate with that information.

10.1 Web Application Framework :

Web Application Framework or essentially Web Framework addresses an assortment of libraries and modules that empowers a web application designer to compose applications without worrying about low-level subtleties, for example, conventions, thread management and so forth.

10.2 Flask

Flask is a web application framework written in Python. It depends on the Werkzeug WSGI toolkit and Jinja2 format motor.

- Web Server Gateway Interface (WSGI) has been received as a norm for Python web application improvement. WSGI is a detail for an all inclusive interface between the web worker and the web applications.
- Werkzeug is a WSGI toolkit which actualizes requests, response objects, and other utility capacities. This empowers assembling a web framework on top of it. The Flask framework utilizes Werkzeug as one of its bases.
- Jinja2 is a famous templating motor for Python. A web templating framework joins a layout with a specific information source to deliver dynamic web pages.

Flask is regularly alluded to as a micro framework. It expects to keep the center of an application basic yet extensible. Flask doesn't have an underlying abstraction layer for handling databases, nor does it have structure and approval upheld. All things being equal, Flask bolsters the augmentations to add such functionality to the application.

10.2.1 Flask Environment

virtualenv is a virtual Python environment builder. It encourages a client to establish numerous Python conditions one next to the other. Consequently, it can evade similarity issues between the various versions of the libraries.

The command that we usually use for creating a virtual environment is :

```
pip install virtualenv
```

10.2.2 Flask Templates

The template files will be put away in the templates index inside the flaskr package. Templates are files that contain static information just as placeholders for dynamic information. A template is delivered with explicit information to create a final record. Flask utilizes the Jinja template library to deliver templates.

In our application, you have utilized templates to deliver HTML which will show in the user's browser. In Flask, Jinja is arranged to autoescape any information that is delivered in HTML templates. This implies that it's protected to deliver client input; any characters they've entered that could meddle with the HTML.

10.2.3 Flask Static Files

Flask consequently adds a static view that takes a path comparative with the flaskr/static directory and serves it. The base.html layout as of now has a connect to the style.css file:

```
{{ url_for('static', filename='style.css') }}
```

A web application frequently requires a static file, for example, a javascript file or a CSS file supporting the presentation of a web page. Typically, the web worker is arranged to serve them for us, yet during the turn of events, these files are served from a static organizer in our package or close to our module and it will be accessible at /static on the application.

An endpoint 'static' is utilized to create URLs for static files.

10.2.4 Sending Form Data To Flask Template

To send form information to the flask template we need to guarantee that the http method can be indicated in the URL rule. Form information obtained by the trigger capacity can be gathered as a word reference item and sent to the template to deliver it on the comparing web page.

```
from flask import Flask, request, render_template
from flask_cors import cross_origin
#import sklearn
#import tensorflow as tf
import numpy as np
#import pandas as pd
#from tensorflow import keras
#from keras.models import Sequential
#from keras.layers import Dense,Activation
from keras.models import load_model


app = Flask(__name__)
model = load_model('fmodel72.h5')



@app.route("/")
@cross_origin()
def home():
    return render_template("home.html")




@app.route("/predict", methods = ["GET", "POST"])
@cross_origin()
def predict():
    if request.method == "POST":
```

10.2.5 Deployment

To switch over from a development environment to an undeniable production environment, an application should be deployed on a genuine web server. Contingent on what we have, there are various choices accessible to deploy a Flask web application.

For little application, we can consider deploying it on any of the accompanying hosted platforms:
Heroku
dotcloud
webfaction

Flask applications can be deployed on these cloud platforms. Likewise, it is conceivable to deploy Flask applications on Google cloud stage. Localtunnel administration permits us to share our application on localhost without meddling with DNS and firewall settings.

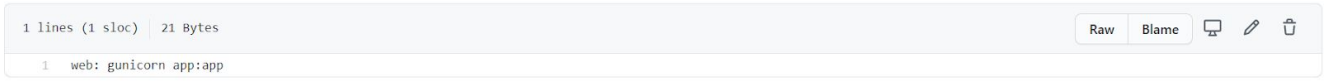
Heroku

Heroku is one of the main cloud platforms as a service (PaaS) and supports several languages. The main thing we need to do is to characterize which libraries our application uses. That way, Heroku knows which ones to accommodate for us, similar to how we install them locally when building up the app. To achieve this, we need to create a requirements.txt file with all of the modules:

```
$ pip freeze > requirements.txt
```

Now that we have our requirements.txt file we also have in it all the libraries that we are utilizing along with their versions.
For Heroku to have the option to run our application like it ought to, we need to characterize a bunch of commands that it should run beforehand.

These commands are located in the Procfile:



The web command advises Heroku to start a web server for the application, utilizing gunicorn.

Steps To Be Followed For Deploying

1. Login to the Heroku account

2. We now have to add our repository to the remote one.

3. After this is done we can now push our project to Heroku using the command :

```
$ git push heroku master
```

4. By doing this we have now successfully uploaded our web app to Heroku.

5. We will find the link for our application under the Setting tab in the Domain and Certificate section.

6. By clicking on this link we can read our application which is now public.

LINK TO OUR PROJECT :

https://accident-severity-pred.herokuapp.com/

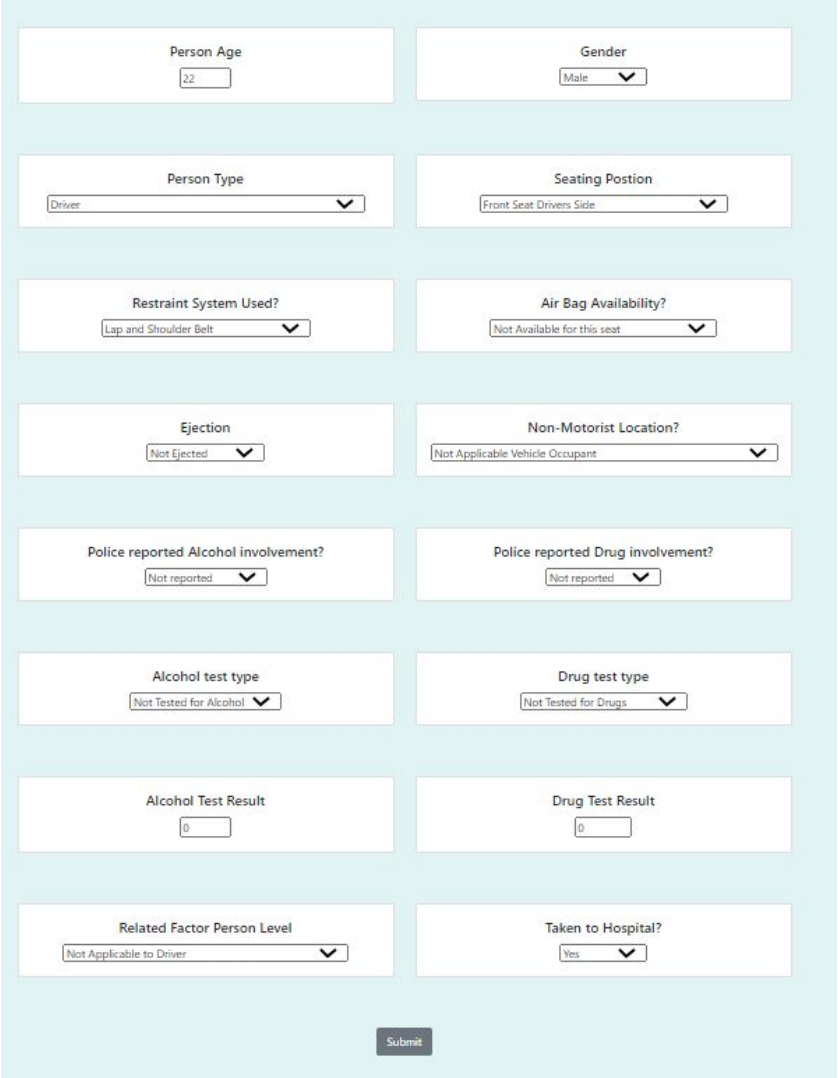


Fig 10.2.5(a)

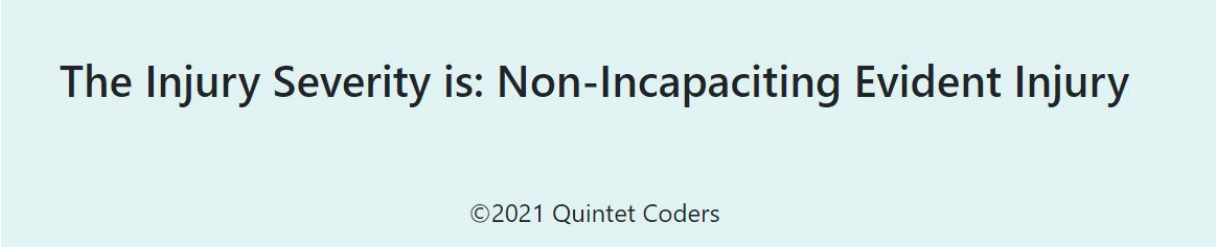


Fig 10.2.5(b)

11. CONCLUSION

The contribution of this study lies in the development of a deep neural network model to analyze traffic accident data and to predict the injury severity of traffic accidents based on traffic accident records collected.

This model encourages in discovering to what in particular level an individual has been harmed in an accident by considering a couple of boundaries and gives us the output dependent on the individual's condition.

For example, No Injury or if the individual has been seriously harmed that it is hard for him/her to live for a more drawn out time then it would foresee the yield to be as Fatal Injury or in situations where the casualty has a couple of light physical wounds that are noticeable then the model predicts the output as NonIncapacitating Evident Injury and correspondingly on account of the individual being harmed however no to the degree of death, the model predicts the output to be Incapacitating Injury. This model moreover would enable us to foresee what stage the injury is at and in like manner avoid potential risk.

12. BIBLIOGRAPHY

- https://machinelearningmastery.com/softmax-activation-function-with-python/
  [Oct 19, 2020]
- https://www.machinecurve.com/index.php/2019/10/06/how-to-use-sparse-categorical-crossentropy-in-keras/
  [6 Oct, 2019]
- https://www.sciencedirect.com/topics/computer-science/neural-network-model
  [Vijay Kotu, Bala Deshpande, in Data Science (Second Edition), 2019]
- https://medium.com/@narkhedesarang?source=post_page-----a9ad42dcfd62----------
  [May 9, 2018]
- https://www.slideshare.net/BIGAKASH/final-year-project-report-35514525
  [Published on Jan 5, 2014]
- https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html

REFERENCES

- [1]https://www.researchgate.net/publication/301708789_Severity_Prediction_of_Traffic_Accident_Using_an_Artificial_Neural_Network_Traffic_Accident_Severity_Prediction_Using_Artificial_Neural_Network
  [Jan 2016, Authors: Sharaf Alkheder, Madhar M. Taamneh, Salah Taamneh]
- [2] Kweon, Y. J., & Kockelman, D. M., Overall Injury Risk to Different Drivers: Combining Exposure, Frequency, and Severity Models. Accident Analysis and Prevention, Vol. 35, 2003, pp. 441-450
  https://www.caee.utexas.edu/prof/kockelman/public_html/TRB02CrashRates.pdf
  [Jan 2002]

- [3] Abdelwahab, H. T. and Abdel-Aty, M. A., Development of Artificial Neural Network Models to Predict Driver Injury Severity in Traffic Accidents at Signalized Intersections. Transportation Research Record 1746, Paper No. 01-2234
  https://journals.sagepub.com/doi/abs/10.3141/1746-02
  [Jan 1, 2001]