**Pain Recognition With Physiological Signals Using Multi-Level Context Information**

**1. INTRODUCTION:**

Pain is the body’s common response to illness that requires medical attention. Traditional pain recognition methods are generally through human observations and subjective recognition. The physiotherapists assess a patient’s pain through exercises during the therapy process and give reasonable exercises to the patient to overcome the disease. Pain recognition depends on the knowledge of each expert, observation, and individual perception through the patient’s expression. This brings many limitations because there are no universal and reliable rules for pain recognition. Therefore, the automation of pain recognition is necessary for humans. In the medical, pain recognition applications is a health monitoring system that helps humans recover from illness through physical therapy exercises. Pain recognition systems use behavior and physiology to perform classification tasks. Measures are physiological signals, facial expressions, body movements, vocalizations, and so on, or a combination of them. In some cases, pain recognition through the patient’s behavior is not reliable. The patient can intentionally control emotional expression. Furthermore, the patients express pain behavior depending on their personality. Some patients lose awareness and do not express painful emotions clearly and reliably. It is difficult to recognize pain through emotional behavior. Therefore, pain recognition using physiological signals is essential. Pain causes the response of the relevant neural structures and alters the measures of differences in physiological signals. Measures of physiological signals related to pain response such as skin conductance, heart rate variability, resting blood pressure, and electroencephalography (EEG). Skin conductance is a signal in response to pain. The increased sympathetic outflow associated with pain secretes the sweat on the skin’s surface. This is the factor to increase electrodermal activity (EDA). The increased sympathetic activity also affects heart rate, increasing heart rate variability or resting blood pressure. In addition, pain affects metabolic areas in the cerebral cortex, or muscle activity . Since the publication of the BioVid Heat Pain Database , EDA and electrocardiogram (ECG) and (electromyogram) EMG signals have become widely used for pain recognition. EDA signals show the skin conductance level, ECG represents the action potential of heart rate and the EMG signal measures muscle activity. The task of automatic pain classification remains extremely challenging. Many previous studies evaluating pain use tools to extract the dynamic characteristic composition of physiological signals to facilitate classification. The efficient methods use representations of physiological signals selected carefully based on medical expertise. The representations include relevant information extracted from the raw data They represent as numeric feature vectors. The set of robust representations can describe the information of the entire data with a size smaller than that of the raw data. These robust representations are fed into inference models and provide fairly high performance. This proves that the skill of selecting representations in machine learning is very necessary for improving the classifier’s performance. It is difficult to take advantage of robust representations because they depend on specialized knowledge of raw datasets. This causes time-consuming self-study and the cost of hiring experts to build robust representations. These studies neglect the powerful automated capabilities of the design model. Deep learning approaches automatically generate suitable representations of raw data. Deep learning architecture is a multi-layer stack of simple modules that can learn and compute non-linear mappings . They entirely replace classical methods and do not depend on specialized knowledge of physiological signals. This study aims to build a deep learning model to replace the conventional methods which rely on expert knowledge of physiological signals. It is possible to eliminate the hand-crafted feature selection carefully. We experiment by extracting contextual representation from physiological signals which have stationary and trending factors. Our idea is to build a contextual representation from the hidden information on a sequence in physiological signals. Contextual representation is the time series characteristics of physiological signals for pain or non-pain manifestations. In this study, the context representations are named multi-level context information. Pain recognition is a binary classification that distinguishes painful and non-painful manifestations. In this work, we evaluate the performance of the proposed model based on Part A of the BioVid Heat Pain Database and the Emopain 2021 dataset. Our method uses simple pre-processed physiological signals that are available in the datasets. Part A of the BioVid Heat Pain database consists of five classes with four painful classes and a baseline class representing a non-painful class. In particular, we perform four classification tasks with each task being a classification between each painful class and baseline class. Our model applies the ability to capture spatial information and reduce spatial resolution while preserving the important characteristics of Convolutional Neural Networks. The model continues to use the Recurrent Neural Network’s ability to extract hidden information. We then propose a combination of multiple levels of context information. As shown in the EDA and ECG illustrations in columns a) and b) of the signals through pain levels affect the stationary and the trending of the time series. Therefore, we choose the combination of EDA and ECG signals without EMG signals. We coordinate multi-level context information from EDA and ECG physiological signals.

**1.1 Objective of the project:**

Automatic pain recognition is essential in healthcare. In previous studies, automatic pain recognition methods preferentially apply the features extracted from physiological signals for conventional models. These methods provide good performance but mainly rely on medical expertise for feature extraction of physiological signals. This paper presents a deep learning approach based on physiological signals that have the role of both feature extraction and classification, regardless of medical expertise. We propose multi-level context information for each physiological signal discriminating between pain and painlessness. Our experimental results prove that multi-level context information performs more significantly than uni-level context information based on Part A of the BioVid Heat Pain database and the Emopain 2021 dataset. For Part A of the BioVid Heat Pain database, our experimental results for pain recognition tasks include Pain 0 and Pain 1, Pain 0 and Pain 2, Pain 0 and Pain 3, and Pain 0 and Pain 4. In the classification task between Pain 0 and Pain 4, the results achieve an average accuracy of 84.8 B1 13.3% for 87 subjects and 87.8 B1 11.4% for 67 subjects in a Leave-One-Subject-Out cross-validation evaluation. The proposed method adopts the ability of deep learning to outperform conventional methods on physiological signals.

**2. LITERATURE SURVEY:**

**‘‘Automatic recognition methods supporting pain assessment: A survey,’’**

Pain is a complex phenomenon, involving sensory and emotional experience, that is often poorly understood, especially in infants, anesthetized patients, and others who cannot speak. Technology supporting pain assessment has the potential to help reduce suffering; however, advances are needed before it can be adopted clinically. This survey paper assesses the state of the art and provides guidance for researchers to help make such advances. First, we overview pain’s biological mechanisms, physiological and behavioral responses, emotional components, as well as assessment methods commonly used in the clinic. Next, we discuss the challenges hampering the development and validation of pain recognition technology, and we survey existing datasets together with evaluation methods. We then present an overview of all automated pain recognition publications indexed in the Web of Science as well as from the proceedings of the major conferences on biomedical informatics and artificial intelligence, to provide understanding of the current advances that have been made. We highlight progress in both non-contact and contact-based approaches, tools using face, voice, physiology, and multi-modal information, the importance of context, and discuss challenges that exist, including identification of ground truth. Finally, we identify underexplored areas such as chronic pain and connections to treatments, and describe promising opportunities for continued advances.

**‘‘The biovid heat pain database data for the advancement and systematic validation of an automated pain recognition system,’’**

The objective measurement of subjective, multi-dimensionally experienced pain is still a problem that has yet to be adequately solved. Though verbal methods (i.e., pain scales, questionnaires) and visual analogue scales are commonly used for measuring clinical pain, they tend to lack in reliability or validity when applied to mentally impaired individuals. Expression of pain and/or its biopotential parameters could represent a solution. While such coding systems already exist, they are both very costly and time-consuming, or have been insufficiently evaluated with regards to the theory of mental tests. Building on the experiences made to date, we collected a database using visual and biopotential signals to advance an automated pain recognition system, to determine its theoretical testing quality, and to optimize its performance. For this purpose, participants were subjected to painful heat stimuli under controlled conditions.

**‘‘The affect move 2021 challenge—Affect recognition from naturalistic movement data,’’**

We ran the first Affective Movement Recognition (AffectMove) challenge that brings together datasets of affective bodily behaviour across different real-life applications to foster work in this area. Research on automatic detection of naturalistic affective body expressions is still lagging behind detection based on other modalities whereas movement behaviour modelling is a very interesting and very relevant research problem for the affective computing community. The AffectMove challenge aimed to take advantage of existing body movement datasets to address key research problems of automatic recognition of naturalistic and complex affective behaviour from this type of data. Participating teams competed to solve at least one of three tasks based on datasets of different sensors types and real-life problems: multimodal EmoPain dataset for chronic pain physical rehabilitation context, we Draw-l Movement dataset for maths problem solving settings, and multimodal Unite-Maastricht Dance dataset. To foster work across datasets, we also challenged participants to take advantage of the data across datasets to improve performances and also test the generalization of their approach across different applications.

**‘‘Methods for person-centered continuous pain intensity assessment from biophysiological channels,’’**

In this work, we present methods for the personalization of a system for the continuous estimation of pain intensity from bio-physiological channels. We investigate various ways to estimate the similarity of persons and to retrieve the most informative ones using meta-information, personality traits, and machine learning techniques. Given this information, specialized classifiers can be created that are both, more efficient in terms of complexity and training times and also more accurate than classifiers trained on the complete data. To capture the most information in the different bio-physiological channels, we cover a broad spectrum of different feature extraction algorithms. Furthermore, we show that the system is capable of running in real-time and discuss issues that arise when dealing with incremental data processing. In extensive experiments we verify the validity of our approach.

**‘‘The use of multiple measurements in taxonomic problems,’’**

The Annals of Human Genetics has an archive of material originally published in print format by the Annals of Eugenics (1925-1954). This material is available in specialised libraries and archives. We believe there is a clear academic interest in making this historical material more widely available to a scholarly audience online. These articles have been made available online, by the Annals of Human Genetics, UCL and Blackwell Publishing Ltd strictly for historical and academic reasons. The work of eugenicists was often pervaded by prejudice against racial, ethnic and disabled groups. Publication of this material online is for scholarly research purposes is not an endorsement or promotion of the views expressed in any of these articles or eugenics in general. All articles are published in full, except where necessary to protect individual privacy. We welcome your comments about this archive and its online publication.

**‘‘XGBoost: A scalable tree boosting system,’’**

Tree boosting is a highly effective and widely used machine learning method. In this paper, we describe a scalable end-to-end tree boosting system called XGBoost, which is used widely by data scientists to achieve state-of-the-art results on many machine learning challenges. We propose a novel sparsity-aware algorithm for sparse data and weighted quantile sketch for approximate tree learning. More importantly, we provide insights on cache access patterns, data compression and sharding to build a scalable tree boosting system. By combining these insights, XGBoost scales beyond billions of examples using far fewer resources than existing systems.

**‘‘Personalized deep bi-LSTM RNN based model for pain intensity classification using EDA signal,’’**

Automatic pain intensity assessment from physiological signals has become an appealing approach, but it remains a largely unexplored research topic. Most studies have used machine learning approaches built on carefully designed features based on the domain knowledge available in the literature on the time series of physiological signals. However, a deep learning framework can automate the feature engineering step, enabling the model to directly deal with the raw input signals for real-time pain monitoring. We investigated a personalized Bidirectional Long short-term memory Recurrent Neural Networks (BiLSTM RNN), and an ensemble of BiLSTM RNN and Extreme Gradient Boosting Decision Trees (XGB) for four-category pain intensity classification. We recorded Electrodermal Activity (EDA) signals from 29 subjects during the cold pressor test. We decomposed EDA signals into tonic and phasic components and augmented them to original signals. The BiLSTM-XGB model outperformed the BiLSTM classification performance and achieved an average F1-score of 0.81 and an Area Under the Receiver Operating Characteristic curve (AUROC) of 0.93 over four pain states: no pain, low pain, medium pain, and high pain. We also explored a concatenation of the deep-learning feature representations and a set of fourteen knowledge-based features extracted from EDA signals. The XGB model trained on this fused feature set showed better performance than when it was trained on component feature sets individually. This study showed that deep learning could let us go beyond expert knowledge and benefit from the generated deep representations of physiological signals for pain assessment.

**3. SYSTEM ANALYSIS**

**3.1 Existing System**

For machine learning, there are many effective methods used in pain recognition. In  the authors explore both video and physiological data, they manually propose features extracted from each signal channel (EMG, ECG, EDA) for pain perception, and their results indicate the Random Forest (RF) model  is effective for automatic pain recognition. Similarly, the authors in perform a broad spectrum of different feature extraction algorithms to extract robust information from EMG, ECG and EDA signals for the RF classifier. In  the authors perform complex signal preprocessing for BVP, ECG and EDA. They then extract statistical features for each of the signal channel. Feature selection and principal component analysis are performed to select high-quality features from the statistical features.

**Disadvantages**

1. Less Secure
2. Low **Detection and Monitoring**

**3.2 Proposed System**

In propose paper author employing multilevel deep learning algorithms for pain recognition as all existing uni-level traditional machine learning algorithms are not good at recognition. Multilevel algorithm giving good accuracy compare to uni-level algorithms like SVM, Random Forest, Linear Regression etc. In propose work author optimizing, selecting features and performing classification using multilevel algorithms like CNN + BI-LSTM (bi-directional LSTM).

**Advantages**

1. More Secure
2. High **Detection and Monitoring**

**3.3. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

**Umbrella Activity**

**Umbrella Activity**

**Umbrella Activity**

1. Feasibility Study
2. TEAM FORMATION
3. Project Specification PREPARATION

Business Requirement Documentation

ANALYSIS & DESIGN

CODE

UNIT TEST

DOCUMENT CONTROL

ASSESSMENT

TRAINING

INTEGRATION & SYSTEM TESTING

DELIVERY/INSTALLATION

ACCEPTANCE TEST

Requirements Gathering

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering** **stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

  
When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artefacts will be produced. Software artefacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artefacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artefact is linked to a specific design element, and that each developed artefact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artefacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artefacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labour data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.4. Software Requirement Specification**

**3.4.1. Overall Description**

A Software Requirements Specification (SRS) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) is a complete description of the behaviour of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analysing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development lifecycle domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* Business requirements describe in business terms what must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:
* **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economic feasibility for certain.

* **Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

* **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to .the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**3.4.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user friendly python Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through python capabilities.

**Software Interfaces**

The required software is python.

**HARDWARE REQUIREMENTS:**

# Processor - I3(min)

* Speed - 1.1 GHz
* RAM - 4GB(min)
* Hard Disk - 500GB

**SOFTWARE REQUIREMENTS:**

* Operating System - Windows 10/above
* Programming Language - Python 3.7 with Jupyter Notebook

**4. SYSTEM DESIGN**

**UML Diagram:**

The Unified Modelling Language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* + **User Model View**
    1. This view represents the system from the users perspective.
    2. The analysis representation describes a usage scenario from the end-users perspective.
  + **Structural Model view**
    1. In this model the data and functionality are arrived from inside the system.
    2. This model view models the static structures.
* **Behavioural Model View**

It represents the dynamic of behavioural as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* **Implementation Model View**

In this the structural and behavioural as parts of the system are represented as they are to be built.

* **Environmental Model View**

In this the structural and behavioural aspects of the environment in which the system is to be implemented are represented.

**Class Diagram:**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake.



**Use case Diagram:**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

****

**Sequence diagram:**

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

****

**Collaboration diagram:**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behaviour of a system.



**Component Diagram:**

In the Unified Modelling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

****

**Deployment Diagram:**

A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

****

**Activity Diagram:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another

Activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent

Upload Dataset

Preprocess Dataset

Calculate Metrices

datasetSplit

datasetSplit

datasetSplit

Run Random Forest

Predict

**Data Flow Diagram:**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

User

1. upload dataset 2.upload dataset successfully

3. preprocess dataset 4.preprocess dataset successfully

5. Calulate Metrices 6. Calculate Metrices Successfully\

7. Run Random Forest Algorithm 8. Random Forest Algorithm Successfully

**5. IMPLEMETATION**

**PYTHON**

**1.1 Introduction**

\* One of the most popular languages is Python. Guido van Rossum released this language in 1991. Python is available on the Mac, Windows, and Raspberry Pi operating systems. The syntax of Python is simple and identical to that of English. When compared to Python, it was seen that the other language requires a few extra lines.

\*It is an interpreter-based language because code may be run line by line after it has been written. This implies that rapid prototyping is possible across all platforms. Python is a big language with a free, binary-distributed interpreter standard library.

\* It is inferior to maintenance that is conducted and is straightforward to learn. It is an object-oriented, interpreted programming language. It supports several different programming paradigms in addition to object-oriented programming, including functional and procedural programming.

\* It supports several different programming paradigms in addition to object-oriented programming, including practical and procedural programming. Python is mighty while maintaining a relatively straightforward syntax. Classes, highly dynamic data types, modules, and exceptions are covered. Python can also be utilised by programmes that require programmable interfaces as an external language.

Here are some key features and characteristics of Python:

* Readability: Python emphasizes code readability with its clean and intuitive syntax. It uses indentation and whitespace to structure code blocks, making it easy to understand and maintain.
* Easy to Learn: Python's simplicity and readability make it an excellent choice for beginners. Its straightforward syntax and extensive documentation make it accessible for newcomers to programming.
* Interpreted Language: Python is an interpreted language, meaning that it doesn't need to be compiled before running. The Python interpreter reads and executes the code directly, making the development process faster and more interactive.
* Cross-platform Compatibility: Python is available for major operating systems like Windows, macOS, and Linux. This cross-platform compatibility allows developers to write code once and run it on different platforms without modifications.
* Large Standard Library: Python comes with a vast standard library that provides ready-to-use modules and functions for various tasks. It covers areas such as file I/O, networking, regular expressions, databases, and more, saving developers time and effort.
* Extensible and Modular: Python supports modular programming, enabling developers to organize code into reusable modules and packages. Additionally, Python allows integrating modules written in other languages, such as C or C++, providing flexibility and performance optimizations.
* Wide Range of Libraries and Frameworks: Python has a vibrant ecosystem with numerous third-party libraries and frameworks. These libraries, such as NumPy, pandas, TensorFlow, and Django, extend Python's capabilities for specific domains, making it a powerful tool for diverse applications.
* Object-Oriented: Python supports object-oriented programming (OOP) principles, allowing developers to create and work with classes and objects. OOP provides a structured approach to code organization, promoting code reuse and modularity.
* Dynamic Typing: Python is dynamically typed, meaning variable types are determined at runtime. Developers do not need to declare variable types explicitly, which enhances flexibility and simplifies code writing.

**1.2 Installation**

To install Python on your computer, follow these basic steps:

* Step 1: Visit the Python website Go to the official Python website at <https://www.python.org/>.
* Step 2: Select the operating system Choose the appropriate installer for your operating system. Python supports Windows, macOS, and various Linux distributions. Make sure to select the correct version that matches your operating system.
* Step 3: Check which version of Python is installed; if the 3.7.0 version is not there, uninstall it through the control panel and
* Step 4: Install Python 3.7.0 using Cmd.
* Step 5: Install the all libraries that required to run the project
* Step 6: Run

**1.3 Python Features:**

1. **Easy:** Because Python is a more accessible and straightforward language, Python programming is easier to learn.
2. **Interpreted language:** Python is an interpreted language, therefore it can be used to examine the code line by line and provide results.
3. **Open Source:** Python is a free online programming language since it is open-source.
4. **Portable:** Python is portable because the same code may be used on several computer standard
5. **libraries:** Python offers a sizable library that we may utilize to create applications quickly.
6. **GUI:** It stands for GUI (Graphical User Interface)
7. **Dynamical typed:** Python is a dynamically typed language, therefore the type of the value will be determined at runtime.

**1.4 Python GUI (Tkinter)**

* Python provides a wide range of options for GUI development (Graphical User Interfaces).
* Tkinter, the most widely used GUI technique, is used for all of them.
* The Tk GUI toolkit offered by Python is used with the conventional Python interface.
* Tkinter is the easiest and quickest way to write Python GUI programs.
* Using Tkinter, creating a GUI is simple.
* A part of Python's built-in library is Tkinter. The GUI programs were created.
* Python and Tkinter together give a straightforward and quick way. The Tk GUI toolkit's object-oriented user interface is called Tkinter.

Making a GUI application is easy using Tkinter. Following are the steps:

1) Install the Tkinter module in place.

2) The GUI application Makes the primary window

3) Include one or more of the widgets mentioned above in the GUI application.

4) Set up the main event loop such that it reacts to each user-initiated event.

Although Tkinter is the only GUI framework included in the Python standard library, Python includes a GUI framework. The default library for Python is called Tkinter. Tk is a scripting language often used in designing, testing, and developing GUIs. Tk is a free, open-source widget toolkit that may be used to build GUI applications in a wide range of computer languages.

**1.5 Python IDLE**

* Python IDLE offers a full-fledged file editor, which gives you the ability to write and execute Python programs from within this program. The built-in file editor also includes several features, like code completion and automatic indentation, that will speed up your coding workflow.
* Guido Van Rossum named Python after the British comedy group Monty Python while the name IDLE was chosen to pay tribute to Eric Idle, who was one of the Monty Python's founding members. IDLE comes bundled with the default implementation of the Python language since the 01.5. 2b1 release
* IDLE is used to execute statements similar to Python Shell. IDLE is used to create, modify, and execute Python code. IDLE provides a fully-featured text editor to write Python scripts and provides features like syntax highlighting, auto-completion, and smart indent.
* IDLE has two modes: interactive and script. We wrote our first program, “Hello, World!” in interactive mode. Interactive mode immediately returns the results of commands you enter into the shell. In script mode, you will write a script and then run it.
* The IDE Python IDLE is a good place to start as it helps you become familiar with the way Python works and understand its syntax. This IDE is good to start programming in Python due to its great debugger, but once you are fluent and start developing projects it is necessary to jump to another, more complete IDE.
* Python IDLE (Integrated Development and Learning Environment) is an interactive development environment included with the Python programming language. It provides a convenient way to write, execute, and debug Python code.

When you install Python, IDLE is typically installed along with it. To open IDLE, you can follow these steps:

* Open the command prompt (Windows) or terminal (macOS/Linux).
* Type "idle" and press Enter. Alternatively, you can specify the version with "idle3" or "idle2" for Python 3 or Python 2, respectively.
* Once IDLE is launched, you will see the Python shell, which is an interactive environment where you can type and execute Python code directly.

Here are some features and functionalities provided by Python IDLE:

* Editor: IDLE includes a text editor where you can write your Python code. It offers syntax highlighting, automatic indentation, and code completion to enhance your coding experience.
* Interactive Shell: The Python shell in IDLE allows you to execute Python code interactively. You can type commands, statements, or function calls directly in the shell, and Python will execute them immediately.
* Debugging: IDLE provides basic debugging capabilities to help you find and fix errors in your code. You can set breakpoints, step through code, inspect variables, and track the program's execution.
* Python Help: IDLE provides access to the Python documentation and built-in help. You can access the help menu to find information about Python modules, functions, classes, and more.
* Script Execution: In addition to the interactive shell, IDLE allows you to run Python scripts stored in files. You can write your code in the editor and execute it as a script to see the output or interact with the program.
* Customization: IDLE can be customized to suit your preferences. You can modify settings related to syntax highlighting, indentation, fonts, and more.
* Python IDLE serves as a beginner-friendly development environment and learning tool. It is suitable for writing small scripts, testing code snippets, experimenting with Python features, and learning the language's basics. However, for more advanced development projects, you may consider using other code editors or integrated development environments (IDEs) that provide additional features and better project management capabilities.

**1.6 Libraries**

In Python, libraries (also referred to as modules or packages) are collections of pre-written code that provide additional functionality and tools to extend the capabilities of the Python language. Libraries contain reusable code that developers can leverage to perform specific tasks without having to write everything from scratch.

Python libraries are designed to solve common problems, such as handling data, performing mathematical operations, interacting with databases, working with files, implementing networking protocols, creating graphical user interfaces (GUIs), and much more. They provide ready-to-use functions, classes, and methods that simplify complex operations and save development time.

**Libraries in Python offer various advantages:**

* Code Reusability:
* Efficiency:
* Collaboration
* Domain-Specific Functionality
* To use a Python library, you need to install it first.

There are some libraries following:

* **Pandas:**

Pandas are a Python computer language library for data analysis and manipulation. It offers a specific operation and data format for handling time series and numerical tables. It differs significantly from the release3-clause of the BSD license. It is a well-liked open-source of opinion that is utilized in machine learning and data analysis.

Pandas are a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Pandas are a Python library used for working with data sets.

* It has functions for analysing, cleaning, exploring, and manipulating data.
* The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.
* Pandas allow us to analyse big data and make conclusions based on statistical theories.
* Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science. Pandas are a Python library for data analysis. Started by Wes McKinney in 2008 out of a need for a powerful and flexible quantitative analysis tool, pandas have grown into one of the most popular Python libraries. It has an extremely active community of contributors. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself.

* **NumPy:**

The NumPy Python library for multi-dimensional, big-scale matrices adds a huge number of high-level mathematical functions. It is possible to modify NumPy by utilizing a Python library. Along with line, algebra, and the Fourier transform operations, it also contains several matrices-related functions.

NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

* NumPy is a Python library used for working with arrays.
* It also has functions for working in domain of linear algebra, Fourier transform, and matrices.
* NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
* NumPy stands for Numerical Python.
* In Python we have lists that serve the purpose of arrays, but they are slow to process.
* NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
* The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.
* Arrays are very frequently used in data science, where speed and resources are very important.
* **Matplotlib:**

It is a multi-platform, array-based data visualization framework built to interact with the whole SciPy stack. MATLAB is proposed as an open-source alternative. Matplotlib is a Python extension and a cross-platform toolkit for graphical plotting and visualization.

Matplotlib is a popular Python library for creating static, animated, and interactive visualizations. It provides a flexible and comprehensive set of tools for generating plots, charts, histograms, scatter plots, and more. Matplotlib is widely used in various fields, including data analysis, scientific research, and data visualization.

Here are some key features and functionalities of the Matplotlib library:

* Plotting Functions
* Customization Options
* Multiple Interfaces
* Integration with NumPy and pandas
* Subplots and Figures:
* Saving and Exporting
* **Scikit-learn:**

The most stable and practical machine learning library for Python is scikit-learn. Regression, dimensionality reduction, classification, and clustering are just a few of the helpful tools it provides through the Python interface for statistical modeling and machine learning. It is an essential part of the Python machine learning toolbox used by JP Morgan. It is frequently used in various machine learning applications, including classification and predictive analysis.

Scikit-learn (also referred to as sklearn) is a widely used open-source machine learning library for Python. It provides a comprehensive set of tools and algorithms for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, model selection, and pre-processing.

Here are some key features and functionalities of the Scikit-learn library:

* Easy-to-Use Interface:
* Broad Range of Algorithms:
* Data Pre-processing and Feature Engineering:
* Model Evaluation and Validation:
* Integration with NumPy and pandas:
* Robust Documentation and Community Support:
* **Keras:**

\* Google's Keras is a cutting-edge deep learning API for creating neural networks. It is created in Python and is designed to simplify the development of neural networks. Additionally, it enables the use of various neural networks for computation. Deep learning models are developed and tested using the free and open-source Python software known as Keras.

Keras is a high-level deep learning library for Python. It is designed to provide a user-friendly and intuitive interface for building and training deep learning models. Keras acts as a front-end API, allowing developers to define and configure neural networks while leveraging the computational backend engines, such as Tensor Flow or Theano.

Here are some key features and functionalities of the Keras library:

* User-Friendly API
* Multi-backend Support
* Wide Range of Neural Network Architectures
* Pre-trained Models and Transfer Learning:
* Easy Model Training and Evaluation:
* GPU Support:
* **h5py:**

\* The h5py Python module offers an interface for the binary HDF5 data format. Thanks to p5py, the top can quickly halt the vast amount of numerical data and alter it using the NumPy library. It employs common syntax for Python, NumPy, and dictionary arrays.

h5py is a Python library that provides a simple and efficient interface for working with datasets and files in the Hierarchical Data Format 5 (HDF5) format. HDF5 is a versatile data format commonly used for storing and managing large volumes of numerical data.

Here are some key features and functionalities of the h5py library:

* + HDF5 File Access
  + Dataset Handling:
  + Group Organization:
  + Attributes:
  + Compatibility with NumPy
  + Performance
* **Tensor flow**

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. TensorFlow is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models.

TensorFlow is a popular open-source library for machine learning and deep learning. It provides a comprehensive set of tools, APIs, and computational resources for building and training various types of machine learning models, especially neural networks.

Here are some key features and functionalities of TensorFlow:

* Neural Network Framework:
* Computational Graphs
* Automatic Differentiation
* GPU and TPU Support
* Distributed Computing
* Deployment Capabilities
* **Tkinter**

Tkinter is an acronym for "Tk interface". Tk was developed as a GUI extension for the Tcl scripting language by John Ousterhout. The first release was in 1991. Tkinter is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library.

Tkinter is a standard Python library used for creating graphical user interfaces (GUIs). It provides a set of modules and classes that allow you to develop interactive and visually appealing desktop applications.

Here are some key features and functionalities of Tkinter:

* Cross-Platform Compatibility
* Simple and Easy-to-Use
* Widgets and Layout Management
* Event-Driven Programming
* Customization and Styling
* Integration with Other Libraries
* **NLTK**

NLTK is a toolkit build for working with NLP in Python. It provides us various text processing libraries with a lot of test datasets. A variety of tasks can be performed using NLTK such as tokenizing, parse tree visualization, etc NLTK (Natural Language Toolkit) is the go-to API for NLP (Natural Language Processing) with Python. It is a really powerful tool to pre-process text data for further analysis like with ML models for instance. It helps convert text into numbers, which the model can then easily work with.

NLTK (Natural Language Toolkit) is a Python library widely used for working with human language data and implementing natural language processing (NLP) tasks. It provides a set of tools, corpora, and resources for tasks such as tokenization, stemming, tagging, parsing, sentiment analysis, and more.

Here are some key features and functionalities of NLTK:

* Text Processing
* Part-of-Speech Tagging
* Named Entity Recognition
* Chunking and Parsing
* Sentiment Analysis:
* WordNet Integration:
* **Scipy**

SciPy is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data.

SciPy is a powerful scientific computing library for Python that provides a wide range of mathematical algorithms and functions. It builds upon NumPy, another fundamental library for numerical computing, and extends its capabilities by adding additional tools for scientific and technical computing tasks.

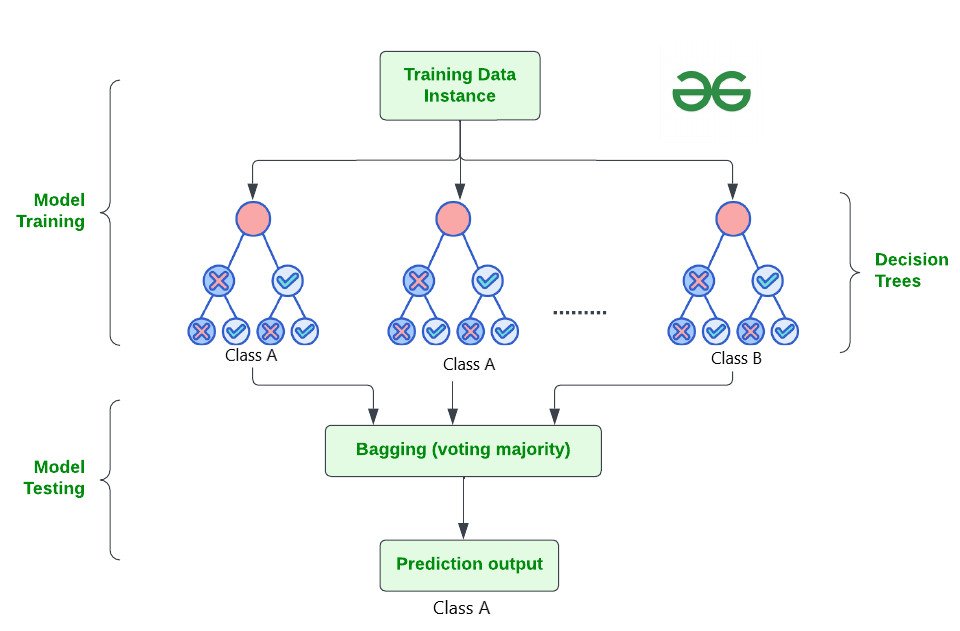
Here are some key features and functionalities of SciPy:

* Numerical Integration:
* Optimization and Root Finding
* Linear Algebra
* Signal and Image Processing
* Statistics

**Random Forest:**

Machine learning, a fascinating blend of computer science and statistics, has witnessed incredible progress, with one standout algorithm being the **Random Forest**. **Random forests or Random Decision Trees** is a collaborative team of **decision trees** that work together to provide a single output. Originating in 2001 through Leo Breiman, Random Forest has become a cornerstone for machine learning enthusiasts. In this article, we will explore the fundamentals and implementation of **Random Forest Algorithm.**

Random Forest algorithm is a powerful tree learning technique in Machine Learning. It works by creating a number of Decision Trees during the training phase. Each tree is constructed using a random subset of the data set to measure a random subset of features in each partition. This randomness introduces variability among individual trees, reducing the risk of overfitting and improving overall prediction performance. In prediction, the algorithm aggregates the results of all trees, either by voting (for classification tasks) or by averaging (for regression tasks) This collaborative decision-making process, supported by multiple trees with their insights, provides an example stable and precise results. Random forests are widely used for classification and regression functions, which are known for their ability to handle complex data, reduce overfitting, and provide reliable forecasts in different environments.



**How Does Random Forest Work?**

The random Forest algorithm works in several steps which are discussed below–>

* **Ensemble of Decision Trees:** Random Forest leverages the power of ensemble learning by constructing an army of Decision Trees. These trees are like individual experts, each specializing in a particular aspect of the data. Importantly, they operate independently, minimizing the risk of the model being overly influenced by the nuances of a single tree.
* **Random Feature Selection:** To ensure that each decision tree in the ensemble brings a unique perspective, Random Forest employs random feature selection. During the training of each tree, a random subset of features is chosen. This randomness ensures that each tree focuses on different aspects of the data, fostering a diverse set of predictors within the ensemble.
* **Bootstrap Aggregating or Bagging:** The technique of bagging is a cornerstone of Random Forest’s training strategy which involves creating multiple bootstrap samples from the original dataset, allowing instances to be sampled with replacement. This results in different subsets of data for each decision tree, introducing variability in the training process and making the model more robust.
* **Decision Making and Voting:** When it comes to making predictions, each decision tree in the Random Forest casts its vote. For classification tasks, the final prediction is determined by the [mode](https://www.geeksforgeeks.org/mode/) (most frequent prediction) across all the trees. In regression tasks, the average of the individual tree predictions is taken. This internal voting mechanism ensures a balanced and collective decision-making process.

**Key Features of Random Forest**

Some of the Key Features of Random Forest are discussed below–>

1. **High Predictive Accuracy:** Imagine Random Forest as a team of decision-making wizards. Each wizard (decision tree) looks at a part of the problem, and together, they weave their insights into a powerful prediction tapestry. This teamwork often results in a more accurate model than what a single wizard could achieve.
2. **Resistance to Overfitting:** Random Forest is like a cool-headed mentor guiding its apprentices (decision trees). Instead of letting each apprentice memorize every detail of their training, it encourages a more well-rounded understanding. This approach helps prevent getting too caught up with the training data which makes the model less prone to overfitting.
3. **Large Datasets Handling:** Dealing with a mountain of data? Random Forest tackles it like a seasoned explorer with a team of helpers (decision trees). Each helper takes on a part of the dataset, ensuring that the expedition is not only thorough but also surprisingly quick.
4. **Variable Importance Assessment:** Think of Random Forest as a detective at a crime scene, figuring out which clues (features) matter the most. It assesses the importance of each clue in solving the case, helping you focus on the key elements that drive predictions.
5. **Built-in Cross-Validation:** Random Forest is like having a personal coach that keeps you in check. As it trains each decision tree, it also sets aside a secret group of cases (out-of-bag) for testing. This built-in validation ensures your model doesn’t just ace the training but also performs well on new challenges.
6. **Handling Missing Values:** Life is full of uncertainties, just like datasets with missing values. Random Forest is the friend who adapts to the situation, making predictions using the information available. It doesn’t get flustered by missing pieces; instead, it focuses on what it can confidently tell us.
7. **Parallelization for Speed:** Random Forest is your time-saving buddy. Picture each decision tree as a worker tackling a piece of a puzzle simultaneously. This parallel approach taps into the power of modern tech, making the whole process faster and more efficient for handling large-scale projects.

## Implement Random Forest for Classification :

# Import necessary libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

import warnings

warnings.filterwarnings('ignore')

# Load the Titanic dataset

url = "https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv"

titanic\_data = pd.read\_csv(url)

# Drop rows with missing target values

titanic\_data = titanic\_data.dropna(subset=['Survived'])

# Select relevant features and target variable

X = titanic\_data[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']]

y = titanic\_data['Survived']

# Convert categorical variable 'Sex' to numerical using .loc

X.loc[:, 'Sex'] = X['Sex'].map({'female': 0, 'male': 1})

# Handle missing values in the 'Age' column using .loc

X.loc[:, 'Age'].fillna(X['Age'].median(), inplace=True)

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create a Random Forest Classifier

rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

# Train the classifier

rf\_classifier.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = rf\_classifier.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

classification\_rep = classification\_report(y\_test, y\_pred)

# Print the results

print(f"Accuracy: {accuracy:.2f}")

print("\nClassification Report:\n", classification\_rep)

**Output:**

Accuracy: 0.80

Classification Report:

precision recall f1-score support

0 0.82 0.85 0.83 105

1 0.77 0.73 0.75 74

accuracy 0.80 179

macro avg 0.79 0.79 0.79 179

weighted avg 0.80 0.80 0.80 179

In the above code, we’re using a Random Forest Classifier to make sense of the Titanic dataset. First, we gather our tools – importing libraries to handle data and evaluate our model. Next, we dive into the Titanic dataset, fixing missing information and choosing important details like a detective solving a mystery. We even teach the computer to understand ‘male’ and ‘female’ by turning them into numbers. Then, we split our dataset into pieces – one part to train our model, and the other to test its newfound skills. Our Random Forest Classifier is like a student, learning from the training set. Once trained, it faces a test – making predictions on the test set. We’re like judges, using a classification report to grade how well our model did.

**Convolutional Neural Network (CNN)**

A **Convolutional Neural Network (CNN)** is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.

When it comes to Machine Learning, Artificial Neural Networks perform really well. Neural Networks are used in various datasets like images, audio, and text. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use **Recurrent Neural Networks** more precisely an LSTM, similarly for image classification we use Convolution Neural networks. In this blog, we are going to build a basic building block for CNN.

In a regular Neural Network there are three types of layers:

1. **Input Layers:** It’s the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).
2. **Hidden Layer:** The input from the Input layer is then fed into the hidden layer. There can be many hidden layers depending on our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of the output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.
3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.

The data is fed into the model and output from each layer is obtained from the above step is called [feedforward](https://www.geeksforgeeks.org/understanding-multi-layer-feed-forward-networks/), we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc. The error function measures how well the network is performing. After that, we backpropagate into the model by calculating the derivatives. This step is called [Backprzopagation](https://www.geeksforgeeks.org/backpropagation-in-data-mining/) which basically is used to minimize the loss.

**Convolution Neural Network**

Convolutional Neural Network (CNN) is the extended version of [artificial neural networks (ANN)](https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/) which is predominantly used to extract the feature from the grid-like matrix dataset. For example visual datasets like images or videos where data patterns play an extensive role.

**CNN architecture**

Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers.

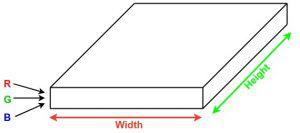


*Simple CNN architecture*

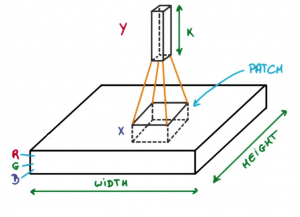
The Convolutional layer applies filters to the input image to extract features, the Pooling layer downsamples the image to reduce computation, and the fully connected layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent.

**How Convolutional Layers works**

Convolution Neural Networks or covnets are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image), and height (i.e the channel as images generally have red, green, and blue channels).



Now imagine taking a small patch of this image and running a small neural network, called a filter or kernel on it, with say, K outputs and representing them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different widths, heights, and depths. Instead of just R, G, and B channels now we have more channels but lesser width and height. This operation is called **Convolution**. If the patch size is the same as that of the image it will be a regular neural network. Because of this small patch, we have fewer weights.



*Image source: Deep Learning Udacity*

Now let’s talk about a bit of mathematics that is involved in the whole convolution process.

* Convolution layers consist of a set of learnable filters (or kernels) having small widths and heights and the same depth as that of input volume (3 if the input layer is image input).

1. For example, if we have to run convolution on an image with dimensions 34x34x3. The possible size of filters can be axax3, where ‘a’ can be anything like 3, 5, or 7 but smaller as compared to the image dimension.
2. During the forward pass, we slide each filter across the whole input volume step by step where each step is called **stride** (which can have a value of 2, 3, or even 4 for high-dimensional images) and compute the dot product between the kernel weights and patch from input volume.
3. As we slide our filters we’ll get a 2-D output for each filter and we’ll stack them together as a result, we’ll get output volume having a depth equal to the number of filters. The network will learn all the filters.

**Layers used to build ConvNets**

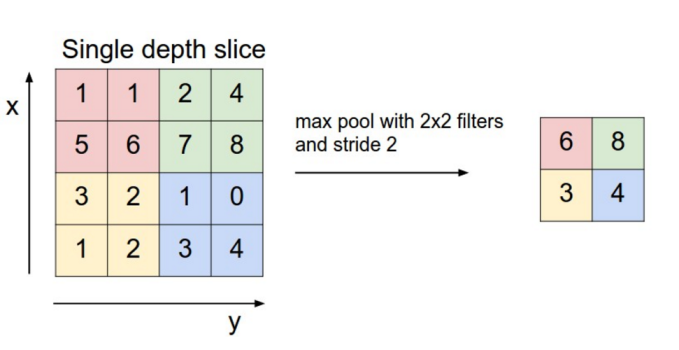
A complete Convolution Neural Networks architecture is also known as covnets. A covnets is a sequence of layers, and every layer transforms one volume to another through a differentiable function.

**Types of layers:**

datasets  
Let’s take an example by running a covnets on of image of dimension 32 x 32 x 3.

* **Input Layers:** It’s the layer in which we give input to our model. In CNN, Generally, the input will be an image or a sequence of images. This layer holds the raw input of the image with width 32, height 32, and depth 3.

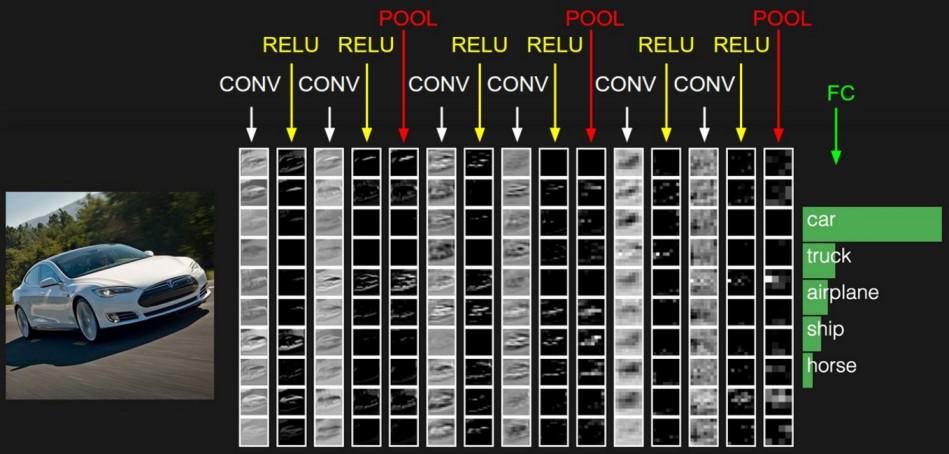
1. **Convolutional Layers:**This is the layer, which is used to extract the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are smaller matrices usually 2×2, 3×3, or 5×5 shape. it slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch. The output of this layer is referred as feature maps. Suppose we use a total of 12 filters for this layer we’ll get an output volume of dimension 32 x 32 x 12.
2. **Activation Layer:**By adding an activation function to the output of the preceding layer, activation layers add nonlinearity to the network. it will apply an element-wise activation function to the output of the convolution layer. Some common activation functions are **RELU**: max(0, x),  **Tanh**, **Leaky RELU**, etc. The volume remains unchanged hence output volume will have dimensions 32 x 32 x 12.
3. **Pooling layer:** This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are **max pooling** and **average pooling**. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.



*Image source: cs231n.stanford.edu*

* **Flattening:**The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.

1. **Fully Connected Layers:**It takes the input from the previous layer and computes the final classification or regression task.



* **Output Layer:** The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class.

**LSTM**

A traditional RNN has a single hidden state that is passed through time, which can make it difficult for the network to learn long-term dependencies. LSTMs address this problem by introducing a memory cell, which is a container that can hold information for an extended period. LSTM networks are capable of learning long-term dependencies in sequential data, which makes them well-suited for tasks such as language translation, speech recognition, and time series forecasting. LSTMs can also be used in combination with other neural network architectures, such as Convolutional Neural Networks (CNNs) for image and video analysis.

The memory cell is controlled by three gates: the input gate, the forget gate, and the output gate. These gates decide what information to add to, remove from, and output from the memory cell. The input gate controls what information is added to the memory cell. The forget gate controls what information is removed from the memory cell. And the output gate controls what information is output from the memory cell. This allows LSTM networks to selectively retain or discard information as it flows through the network, which allows them to learn long-term dependencies.

**Bidirectional LSTM**

Bidirectional LSTM (Bi LSTM/ BLSTM) is recurrent neural network (RNN) that is able to process sequential data in both forward and backward directions. This allows Bi LSTM to learn longer-range dependencies in sequential data than traditional LSTMs, which can only process sequential data in one direction.

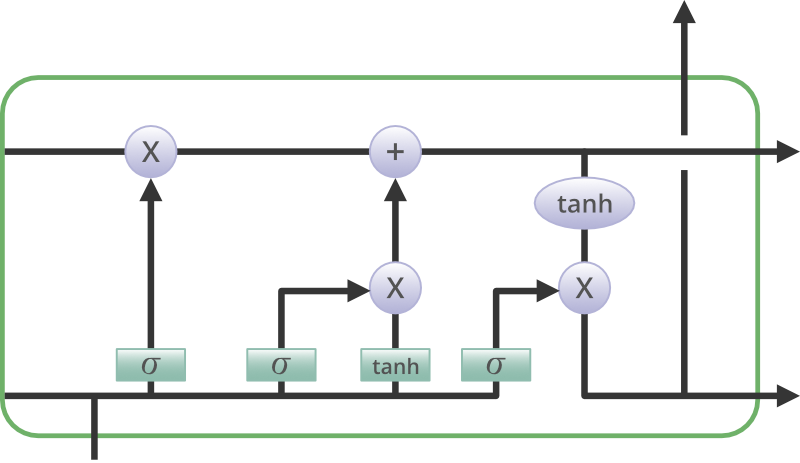
* Bi LSTMs are made up of two LSTM networks, one that processes the input sequence in the forward direction and one that processes the input sequence in the backward direction. The outputs of the two LSTM networks are then combined to produce the final output.

1. Bi LSTM have been shown to achieve state-of-the-art results on a wide variety of tasks, including machine translation, speech recognition, and text summarization.

LSTMs can be stacked to create deep LSTM networks, which can learn even more complex patterns in sequential data. Each LSTM layer captures different levels of abstraction and temporal dependencies in the input data.

**Architecture and Working of LSTM**

LSTM architecture has a chain structure that contains four neural networks and different memory blocks called **cells**.



Information is retained by the cells and the memory manipulations are done by the**gates.** There are three gates –

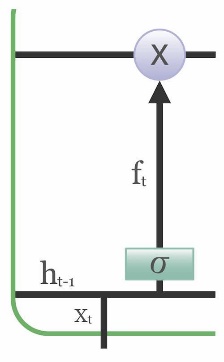
**Forget Gate**

The information that is no longer useful in the cell state is removed with the forget gate. Two inputs *xt* (input at the particular time) and *ht-1* (previous cell output) are fed to the gate and multiplied with weight matrices followed by the addition of bias. The resultant is passed through an activation function which gives a binary output. If for a particular cell state the output is 0, the piece of information is forgotten and for output 1, the information is retained for future use. The equation for the forget gate is:

 where:

* W\_f represents the weight matrix associated with the forget gate.

1. [h\_t-1, x\_t] denotes the concatenation of the current input and the previous hidden state.
2. b\_f is the bias with the forget gate.
3. σ is the sigmoid activation function.



**Input gate**

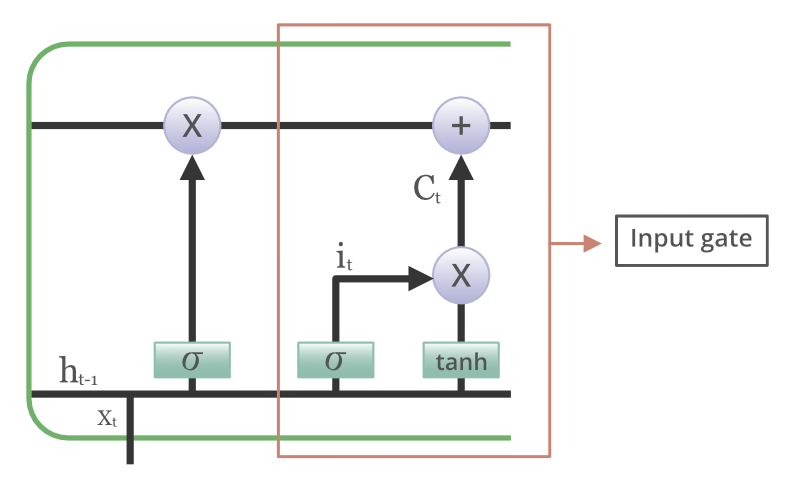
The addition of useful information to the cell state is done by the input gate. First, the information is regulated using the sigmoid function and filter the values to be remembered similar to the forget gate using inputs*ht-1* and *xt*. . Then, a vector is created using*tanh*function that gives an output from -1 to +1, which contains all the possible values from ht-1 and *xt*. At last, the values of the vector and the regulated values are multiplied to obtain the useful information. The equation for the input gate is:

We multiply the previous state by ft, disregarding the information we had previously chosen to ignore. Next, we include it∗Ct. This represents the updated candidate values, adjusted for the amount that we chose to update each state value.

where

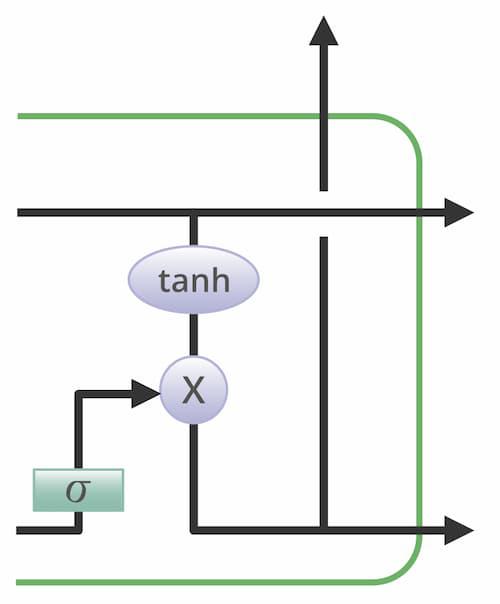
* ⊙ denotes element-wise multiplication

1. tanh is tanh activation function



**Output gate**

The task of extracting useful information from the current cell state to be presented as output is done by the output gate. First, a vector is generated by applying tanh function on the cell. Then, the information is regulated using the sigmoid function and filter by the values to be remembered using inputs*ht-1* and *xt*. At last, the values of the vector and the regulated values are multiplied to be sent as an output and input to the next cell. The equation for the output gate is:



# Gated Recurrent Unit Networks

Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN) that was introduced by Cho et al. in 2014 as a simpler alternative to Long Short-Term Memory (LSTM) networks. Like LSTM, GRU can process sequential data such as text, speech, and time-series data.

The basic idea behind GRU is to use gating mechanisms to selectively update the hidden state of the network at each time step. The gating mechanisms are used to control the flow of information in and out of the network. The GRU has two gating mechanisms, called the reset gate and the update gate.

The reset gate determines how much of the previous hidden state should be forgotten, while the update gate determines how much of the new input should be used to update the hidden state. The output of the GRU is calculated based on the updated hidden state.

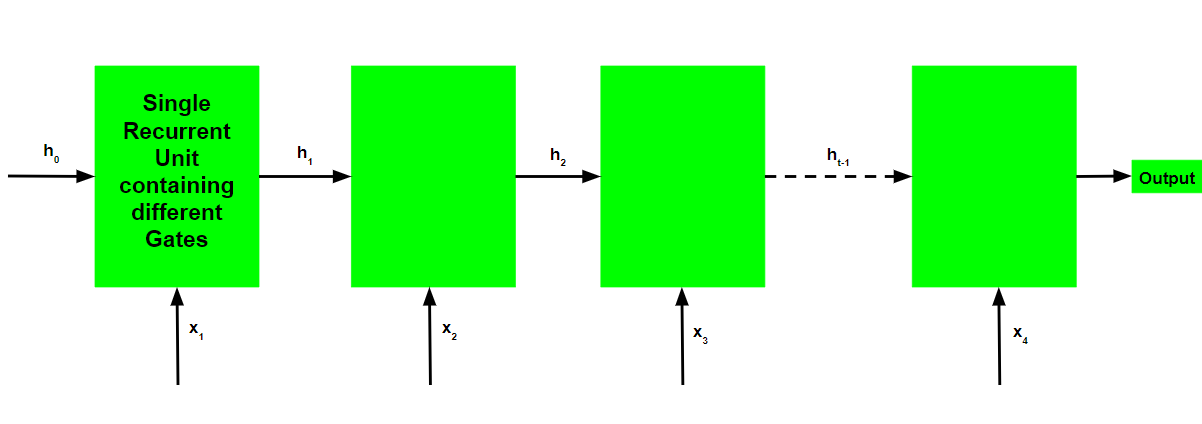
The equations used to calculate the reset gate, update gate, and hidden state of a GRU are as follows:

In summary, GRU networks are a type of RNN that use gating mechanisms to selectively update the hidden state at each time step, allowing them to effectively model sequential data. They have been shown to be effective in various natural language processing tasks, such as language modeling, machine translation, and speech recognition

**Prerequisites: Recurrent Neural Networks, Long Short Term Memory Networks**   
  
To solve the Vanishing-Exploding gradients problem often encountered during the operation of a basic Recurrent Neural Network, many variations were developed. One of the most famous variations is the **Long Short Term Memory Network(LSTM)**. One of the lesser-known but equally effective variations is the **Gated Recurrent Unit Network(GRU)**.   
  
Unlike LSTM, it consists of only three gates and does not maintain an Internal Cell State. The information which is stored in the Internal Cell State in an LSTM recurrent unit is incorporated into the hidden state of the Gated Recurrent Unit. This collective information is passed onto the next Gated Recurrent Unit. The different gates of a GRU are as described below:-

1. **Update Gate(z):** It determines how much of the past knowledge needs to be passed along into the future. It is analogous to the Output Gate in an LSTM recurrent unit.
2. **Reset Gate(r):** It determines how much of the past knowledge to forget. It is analogous to the combination of the Input Gate and the Forget Gate in an LSTM recurrent unit.
3. **Current Memory Gate():** It is often overlooked during a typical discussion on Gated Recurrent Unit Network. It is incorporated into the Reset Gate just like the Input Modulation Gate is a sub-part of the Input Gate and is used to introduce some non-linearity into the input and to also make the input Zero-mean. Another reason to make it a sub-part of the Reset gate is to reduce the effect that previous information has on the current information that is being passed into the future.

The basic work-flow of a Gated Recurrent Unit Network is similar to that of a basic Recurrent Neural Network when illustrated, the main difference between the two is in the internal working within each recurrent unit as Gated Recurrent Unit networks consist of gates which modulate the current input and the previous hidden state.



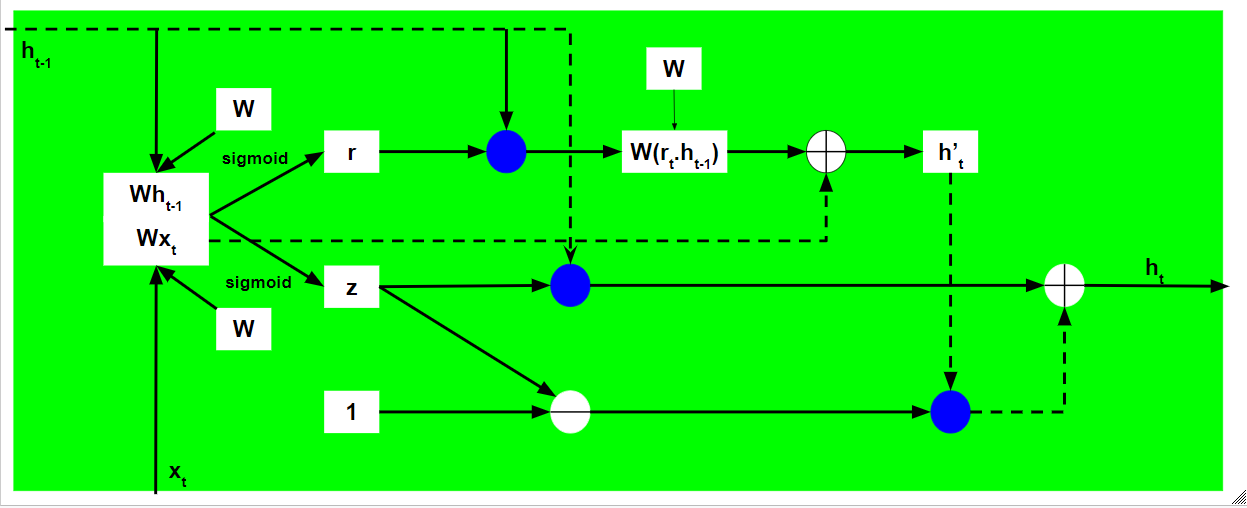
**Working of a Gated Recurrent Unit:**

* Take input the current input and the previous hidden state as vectors.
* Calculate the values of the three different gates by following the steps given below:-   
  1. For each gate, calculate the parameterized current input and previously hidden state vectors by performing element-wise multiplication (Hadamard Product) between the concerned vector and the respective weights for each gate.
  2. Apply the respective activation function for each gate element-wise on the parameterized vectors. Below given is the list of the gates with the activation function to be applied for the gate.

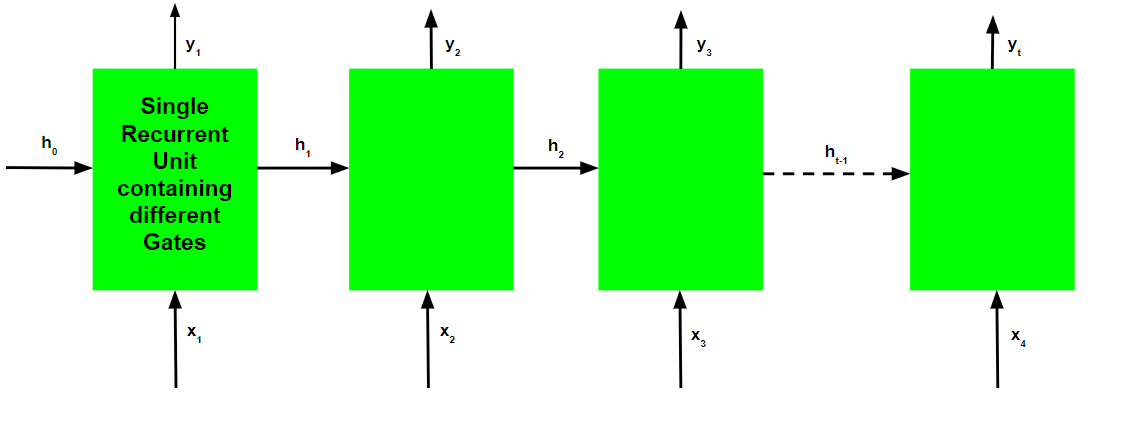
Update Gate : Sigmoid Function

Reset Gate : Sigmoid Function

* The process of calculating the Current Memory Gate is a little different. First, the Hadamard product of the Reset Gate and the previously hidden state vector is calculated. Then this vector is parameterized and then added to the parameterized current input vector.
* To calculate the current hidden state, first, a vector of ones and the same dimensions as that of the input is defined. This vector will be called ones and mathematically be denoted by 1. First, calculate the Hadamard Product of the update gate and the previously hidden state vector. Then generate a new vector by subtracting the update gate from ones and then calculate the Hadamard Product of the newly generated vector with the current memory gate. Finally, add the two vectors to get the currently hidden state vector.  
    
    
  The above-stated working is stated as below:-



Note that the blue circles denote element-wise multiplication. The positive sign in the circle denotes vector addition while the negative sign denotes vector subtraction(vector addition with negative value). The weight matrix W contains different weights for the current input vector and the previous hidden state for each gate.   
Just like Recurrent Neural Networks, a GRU network also generates an output at each time step and this output is used to train the network using gradient descent.



Note that just like the workflow, the training process for a GRU network is also diagrammatically similar to that of a basic Recurrent Neural Network and differs only in the internal working of each recurrent unit.   
The Back-Propagation Through Time Algorithm for a Gated Recurrent Unit Network is similar to that of a Long Short Term Memory Network and differs only in the differential chain formation. Let be the predicted output at each time step and be the actual output at each time step. Then the error at each time step is given by:-   
The total error is thus given by the summation of errors at all time steps. Similarly, the value can be calculated as the summation of the gradients at each time step. Using the chain rule and using the fact that is a function of and which indeed is function of , the following expression arises:-   
Thus the total error gradient is given by the following:-   
Note that the gradient equation involves a chain of which looks similar to that of a basic Recurrent Neural Network but this equation works differently because of the internal workings of the derivatives of .   
**How do Gated Recurrent Units solve the problem of vanishing gradients?**

The value of the gradients is controlled by the chain of derivatives starting from . Recall the expression for :-

Using the above expression, the value for is:-   
Recall the expression for :-

Using the above expression to calculate the value of :-   
Since both the update and reset gate use the sigmoid function as their activation function, both can take values either 0 or 1.   
  
**Case 1(z = 1):** In this case, irrespective of the value of , the term is equal to z which in turn is equal to 1.   
  
**Case 2A(z=0 and r=0):** In this case, the term is equal to 0.

**Case 2B(z=0 and r=1):** In this case, the term is equal to . This value is controlled by the weight matrix which is trainable and thus the network learns to adjust the weights in such a way that the term comes closer to 1.   
  
Thus the Back-Propagation Through Time algorithm adjusts the respective weights in such a manner that the value of the chain of derivatives is as close to 1 as possible.

**Sample Code:**

#importing require python classes and packages  
import seaborn as sns  
from sklearn.metrics import precision\_score  
from sklearn.metrics import recall\_score  
from sklearn.metrics import f1\_score  
from sklearn.metrics import confusion\_matrix  
import matplotlib.pyplot as plt  
import pickle  
  
import numpy as np  
import pandas as pd  
from sklearn.metrics import accuracy\_score  
from sklearn.preprocessing import MinMaxScaler  
from sklearn.model\_selection import train\_test\_split  
  
from keras.models import Sequential, load\_model  
from keras.layers import Dense, TimeDistributed, Conv1D, MaxPooling1D, Flatten, Activation, RepeatVector  
from keras.layers import LSTM #class for LSTM training  
import os  
from keras.layers import Dropout  
from keras.callbacks import ModelCheckpoint  
from keras.layers import Bidirectional, GRU #class for bidirectional LSTM as BILSTM and GRU  
from keras.utils.np\_utils import to\_categorical  
  
from sklearn.metrics import accuracy\_score  
import pickle  
from sklearn import svm  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.model\_selection import LeaveOneOut #class to calculate LOSO  
from sklearn.model\_selection import cross\_val\_score

Using TensorFlow backend.  
c:\users\user\appdata\local\programs\python\python37\lib\site-packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.  
 \_np\_qint8 = np.dtype([("qint8", np.int8, 1)])  
c:\users\user\appdata\local\programs\python\python37\lib\site-packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.  
 \_np\_quint8 = np.dtype([("quint8", np.uint8, 1)])  
c:\users\user\appdata\local\programs\python\python37\lib\site-packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.  
 \_np\_qint16 = np.dtype([("qint16", np.int16, 1)])  
c:\users\user\appdata\local\programs\python\python37\lib\site-packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.  
 \_np\_quint16 = np.dtype([("quint16", np.uint16, 1)])  
c:\users\user\appdata\local\programs\python\python37\lib\site-packages\tensorflow\python\framework\dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.  
 \_np\_qint32 = np.dtype([("qint32", np.int32, 1)])  
c:\users\user\appdata\local\programs\python\python37\lib\site-packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.  
 np\_resource = np.dtype([("resource", np.ubyte, 1)])  
c:\users\user\appdata\local\programs\python\python37\lib\site-packages\tensorboard\compat\tensorflow\_stub\dtypes.py:541: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.  
 \_np\_qint8 = np.dtype([("qint8", np.int8, 1)])  
c:\users\user\appdata\local\programs\python\python37\lib\site-packages\tensorboard\compat\tensorflow\_stub\dtypes.py:542: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.  
 \_np\_quint8 = np.dtype([("quint8", np.uint8, 1)])  
c:\users\user\appdata\local\programs\python\python37\lib\site-packages\tensorboard\compat\tensorflow\_stub\dtypes.py:543: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.  
 \_np\_qint16 = np.dtype([("qint16", np.int16, 1)])  
c:\users\user\appdata\local\programs\python\python37\lib\site-packages\tensorboard\compat\tensorflow\_stub\dtypes.py:544: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.  
 \_np\_quint16 = np.dtype([("quint16", np.uint16, 1)])  
c:\users\user\appdata\local\programs\python\python37\lib\site-packages\tensorboard\compat\tensorflow\_stub\dtypes.py:545: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.  
 \_np\_qint32 = np.dtype([("qint32", np.int32, 1)])  
c:\users\user\appdata\local\programs\python\python37\lib\site-packages\tensorboard\compat\tensorflow\_stub\dtypes.py:550: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.  
 np\_resource = np.dtype([("resource", np.ubyte, 1)])

#use to scale or normalize dataset values  
scaler = MinMaxScaler((0,1))

#load and display dataset values  
dataset = pd.read\_csv("Dataset/BioVid\_coords.csv")  
dataset.fillna(0, inplace = True)  
dataset

0 1 2 3 4 5 6 7 8 9 \  
0 0.0 169.0 133.0 169.0 146.0 171.0 160.0 173.0 172.0 177.0   
1 0.0 169.0 131.0 170.0 145.0 171.0 158.0 173.0 171.0 177.0   
2 0.0 170.0 132.0 170.0 145.0 172.0 158.0 173.0 171.0 177.0   
3 0.0 170.0 131.0 170.0 145.0 172.0 158.0 174.0 172.0 178.0   
4 0.0 169.0 131.0 170.0 145.0 172.0 158.0 174.0 171.0 177.0   
... ... ... ... ... ... ... ... ... ... ...   
7999 57.0 170.0 136.0 170.0 149.0 172.0 162.0 173.0 174.0 176.0   
8000 57.0 171.0 135.0 171.0 148.0 172.0 161.0 174.0 174.0 176.0   
8001 57.0 171.0 135.0 171.0 148.0 172.0 161.0 174.0 174.0 176.0   
8002 57.0 171.0 136.0 171.0 149.0 172.0 162.0 173.0 174.0 176.0   
8003 57.0 171.0 136.0 171.0 150.0 172.0 162.0 173.0 175.0 176.0   
  
 ... 127 128 129 130 131 132 133 134 135 \  
0 ... 228.0 183.0 238.0 183.0 228.0 183.0 223.0 184.0 218.0   
1 ... 228.0 183.0 238.0 184.0 228.0 184.0 223.0 185.0 218.0   
2 ... 228.0 182.0 238.0 184.0 228.0 187.0 223.0 187.0 218.0   
3 ... 228.0 182.0 238.0 184.0 228.0 187.0 223.0 187.0 218.0   
4 ... 228.0 183.0 237.0 184.0 228.0 186.0 223.0 187.0 218.0   
... ... ... ... ... ... ... ... ... ... ...   
7999 ... 225.0 189.0 234.0 190.0 224.0 189.0 219.0 189.0 215.0   
8000 ... 224.0 189.0 234.0 190.0 224.0 189.0 219.0 189.0 215.0   
8001 ... 225.0 189.0 234.0 190.0 224.0 189.0 219.0 189.0 215.0   
8002 ... 224.0 189.0 234.0 190.0 224.0 189.0 219.0 189.0 215.0   
8003 ... 224.0 189.0 234.0 190.0 224.0 189.0 219.0 189.0 214.0   
  
 Label   
0 0   
1 0   
2 0   
3 0   
4 0   
... ...   
7999 3   
8000 3   
8001 3   
8002 3   
8003 3   
  
[8004 rows x 137 columns]

data = dataset.values  
plt.plot(data)   
plt.xlabel("Number of Records")  
plt.ylabel("Signals")  
plt.title("EEG Signal from all Subjects")  
plt.show()

#plot labels in dataset  
labels, count = np.unique(dataset['Label'], return\_counts = True)  
labels = ["Pain0", "Pain1", "Pain2" ,"Pain3", "Pain4"]  
height = count  
bars = labels  
y\_pos = np.arange(len(bars))  
plt.bar(y\_pos, height)  
plt.xticks(y\_pos, bars)  
plt.xlabel("Dataset Class Label Graph")  
plt.ylabel("Count")  
plt.show()

#code to extract X training features and Y label from dataset and then normalize and shuffle dataset values  
dataset = dataset.values  
X = dataset[:,0:dataset.shape[1]-1] #extract training features as X  
Y = dataset[:,dataset.shape[1]-1] #extract target pain label  
X = scaler.fit\_transform(X)#normalized features  
indices = np.arange(X.shape[0])  
np.random.shuffle(indices) #shuffle the dataset values  
X = X[indices]  
Y = Y[indices]  
print("Normalized Training Features : "+str(X))

Normalized Training Features : [[0.19298246 0.45454545 0.63636364 ... 0.5 0.38461538 0.58333333]  
 [0.50877193 0.27272727 0.63636364 ... 0.41666667 0.38461538 0.41666667]  
 [0.07017544 0.54545455 0.36363636 ... 0.41666667 0.53846154 0.41666667]  
 ...  
 [0.8245614 0.27272727 0.54545455 ... 0.25 0.46153846 0.25 ]  
 [0.92982456 0.27272727 0.63636364 ... 0.25 0.38461538 0.25 ]  
 [0.75438596 0.81818182 0.09090909 ... 0.91666667 0.07692308 0.91666667]]

#split dataset into train and test where 80% dataset is for training and 20 for testing  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2) #split dataset into train and test  
print()  
print("Dataset train & test split as 80% dataset for training and 20% for testing")  
print("Training Size (80%): "+str(X\_train.shape[0])) #print training and test size  
print("Testing Size (20%): "+str(X\_test.shape[0]))  
print()

Dataset train & test split as 80% dataset for training and 20% for testing  
Training Size (80%): 6403  
Testing Size (20%): 1601

#define global variables to calculate and store accuracy and other metrics  
precision = []  
recall = []  
fscore = []  
accuracy = []  
loso = []

#function to calculate various metrics such as accuracy, precision etc  
def calculateMetrics(algorithm, predict, testY, loso\_value):  
 p = precision\_score(testY, predict,average='macro') \* 100  
 r = recall\_score(testY, predict,average='macro') \* 100  
 f = f1\_score(testY, predict,average='macro') \* 100  
 a = accuracy\_score(testY,predict)\*100   
 print()  
 print(algorithm+' Accuracy : '+str(a))  
 print(algorithm+' Precision : '+str(p))  
 print(algorithm+' Recall : '+str(r))  
 print(algorithm+' FMeasure : '+str(f))   
 print(algorithm+' LOSO : '+str(loso\_value))  
 loso.append(loso\_value)  
 accuracy.append(a)  
 precision.append(p)  
 recall.append(r)  
 fscore.append(f)  
 conf\_matrix = confusion\_matrix(testY, predict)   
 plt.figure(figsize =(5, 5))   
 ax = sns.heatmap(conf\_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="viridis" ,fmt ="g");  
 ax.set\_ylim([0,len(labels)])  
 plt.title(algorithm+" Confusion matrix")   
 plt.ylabel('True class')   
 plt.xlabel('Predicted class')   
 plt.show()

#train existing Random Forest algorithm and then calculate LOSO and other metrics  
rf = RandomForestClassifier(ccp\_alpha=0.2)  
rf.fit(X\_train, y\_train)#train random forest algorithm  
predict = rf.predict(X\_test)#perform prediction on test data  
cv = LeaveOneOut() #calculate leave one out as LOSO  
loso\_score = cross\_val\_score(rf, X\_test, y\_test, scoring='f1\_micro', cv=cv, n\_jobs=-1)  
calculateMetrics("Existing Random Forest", predict, y\_test, np.mean(loso\_score))#call function to calculate accuracy and other metrics

c:\users\user\appdata\local\programs\python\python37\lib\site-packages\sklearn\metrics\\_classification.py:1272: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))

Existing Random Forest Accuracy : 53.52904434728295  
Existing Random Forest Precision : 27.39956075822512  
Existing Random Forest Recall : 39.59266802443992  
Existing Random Forest FMeasure : 31.01537571578465  
Existing Random Forest LOSO : 0.5352904434728295

#now train propose CNN + BILSTM algorithm on training features  
#reshape training data  
X\_train = np.reshape(X\_train, (X\_train.shape[0], 34, 4))  
X\_test = np.reshape(X\_test, (X\_test.shape[0], 34, 4))  
y\_train = to\_categorical(y\_train)  
y\_test = to\_categorical(y\_test)  
#create CNN sequential object  
propose\_model = Sequential()  
#create CNN1D layer with 32 neurons for data filteration and pool size as 3  
propose\_model.add(Conv1D(filters=32, kernel\_size = 3, activation = 'relu', input\_shape = (X\_train.shape[1], X\_train.shape[2])))  
#defining another CNN layer with 64 neurons  
propose\_model.add(Conv1D(filters=64, kernel\_size = 2, activation = 'relu'))  
propose\_model.add(Conv1D(filters=128, kernel\_size = 2, activation = 'relu'))  
#max pooling layer to collect relevant features from CNN layer  
propose\_model.add(MaxPooling1D(pool\_size = 1))  
propose\_model.add(Flatten())  
propose\_model.add(RepeatVector(2))  
#defining BILSTM kayer with 32 neurons to optimize CNN features  
propose\_model.add(Bidirectional(LSTM(32, activation = 'relu', return\_sequences=True)))  
propose\_model.add(Bidirectional(LSTM(64, activation = 'relu')))  
#adding dropout layer to remove irrelevant features  
propose\_model.add(Dropout(0.2))  
#defining output an dprediction layer  
propose\_model.add(Dense(units = 100, activation = 'softmax'))  
propose\_model.add(Dense(units = y\_train.shape[1], activation = 'softmax'))  
#train and compile the model  
propose\_model.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])  
if os.path.exists("model/propose\_weights.hdf5") == False:  
 model\_check\_point = ModelCheckpoint(filepath='model/propose\_weights.hdf5', verbose = 1, save\_best\_only = True)  
 hist = propose\_model.fit(X\_train, y\_train, batch\_size = 32, epochs = 10, validation\_data=(X\_test, y\_test), callbacks=[model\_check\_point], verbose=1)  
 f = open('model/propose\_history.pckl', 'wb')  
 pickle.dump(hist.history, f)  
 f.close()   
else:  
 propose\_model = load\_model("model/propose\_weights.hdf5")  
#perform prediction on test data   
predict = propose\_model.predict(X\_test)  
predict = np.argmax(predict, axis=1)  
y\_test1 = np.argmax(y\_test, axis=1)  
#calculate LOSO score  
loso\_score = np.amax(propose\_model.evaluate(X\_test, y\_test, batch\_size=16))  
calculateMetrics("Propose CNN + BILSTM", predict, y\_test1, loso\_score)#call function to calculate accuracy and other metrics

WARNING:tensorflow:From c:\users\user\appdata\local\programs\python\python37\lib\site-packages\keras\backend\tensorflow\_backend.py:4070: The name tf.nn.max\_pool is deprecated. Please use tf.nn.max\_pool2d instead.  
  
WARNING:tensorflow:From c:\users\user\appdata\local\programs\python\python37\lib\site-packages\keras\backend\tensorflow\_backend.py:422: The name tf.global\_variables is deprecated. Please use tf.compat.v1.global\_variables instead.  
  
1601/1601 [==============================] - 1s 913us/step  
  
Propose CNN + BILSTM Accuracy : 85.63397876327295  
Propose CNN + BILSTM Precision : 87.88368403021943  
Propose CNN + BILSTM Recall : 80.66030205892093  
Propose CNN + BILSTM FMeasure : 74.77762722412933  
Propose CNN + BILSTM LOSO : 0.8563398122787476

#create extension model using CNN1D + BILSTM + GRU as each algorithm has its own implementation of fetaures extraction so  
#BILSTM will extract optimize features from CNN and then GRU will extract features BILSTM so will have features after 3  
#optimization algorithm so will get best accuracy  
extension\_model = Sequential()  
#create CNN1D layer with 32 neurons for data filteration and pool size as 3  
extension\_model.add(Conv1D(filters=32, kernel\_size = 3, activation = 'relu', input\_shape = (X\_train.shape[1], X\_train.shape[2])))  
extension\_model.add(Conv1D(filters=64, kernel\_size = 2, activation = 'relu'))  
extension\_model.add(Conv1D(filters=128, kernel\_size = 2, activation = 'relu'))  
extension\_model.add(MaxPooling1D(pool\_size = 1))  
extension\_model.add(Flatten())  
extension\_model.add(RepeatVector(2))  
#adding LSTM Bidirectional layer to obtained optimized features from CNN  
extension\_model.add(Bidirectional(LSTM(32, activation = 'relu', return\_sequences=True)))  
#now bidirectional GRU will extract optimized fetaures from BI-LSTM and then train a model with below prediction layer  
extension\_model.add(Bidirectional(GRU(64, activation = 'relu')))  
extension\_model.add(Dropout(0.2))  
#Define output prediction layer  
extension\_model.add(Dense(units = 100, activation = 'softmax'))  
extension\_model.add(Dense(units = y\_train.shape[1], activation = 'softmax'))  
#compile and train the model  
extension\_model.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])  
if os.path.exists("model/extension\_weights.hdf5") == False:  
 model\_check\_point = ModelCheckpoint(filepath='model/extension\_weights.hdf5', verbose = 1, save\_best\_only = True)  
 hist = extension\_model.fit(X\_train, y\_train, batch\_size = 32, epochs = 10, validation\_data=(X\_test, y\_test), callbacks=[model\_check\_point], verbose=1)  
 f = open('model/extension\_history.pckl', 'wb')  
 pickle.dump(hist.history, f)  
 f.close()   
else:  
 extension\_model = load\_model("model/extension\_weights.hdf5")  
#perform prediction on test data   
predict = extension\_model.predict(X\_test)  
predict = np.argmax(predict, axis=1)  
y\_test1 = np.argmax(y\_test, axis=1)  
#calculate LOSO score  
loso\_score = np.amax(extension\_model.evaluate(X\_test, y\_test, batch\_size=16))  
calculateMetrics("Propose CNN + BILSTM", predict, y\_test1, loso\_score)#call function to calculate accuracy and other metrics

1601/1601 [==============================] - 1s 891us/step  
  
Propose CNN + BILSTM Accuracy : 98.50093691442848  
Propose CNN + BILSTM Precision : 98.6004113128408  
Propose CNN + BILSTM Recall : 98.79487679686075  
Propose CNN + BILSTM FMeasure : 98.68628535006435  
Propose CNN + BILSTM LOSO : 0.9850093722343445

#all algorithms performance graph  
df = pd.DataFrame([['Existing Random Forest','Precision',precision[0]],['Existing Random Forest','Recall',recall[0]],['Existing Random Forest','F1 Score',fscore[0]],['Existing Random Forest','Accuracy',accuracy[0]],  
 ['Propsoe CNN + BI-LSTM','Precision',precision[1]],['Propsoe CNN + BI-LSTM','Recall',recall[1]],['Propsoe CNN + BI-LSTM','F1 Score',fscore[1]],['Propsoe CNN + BI-LSTM','Accuracy',accuracy[1]],  
 ['Extension CNN + BI-LSTM + BI-GRU','Precision',precision[2]],['Extension CNN + BI-LSTM + BI-GRU','Recall',recall[2]],['Extension CNN + BI-LSTM + BI-GRU','F1 Score',fscore[2]],['Extension CNN + BI-LSTM + BI-GRU','Accuracy',accuracy[2]],  
 ],columns=['Parameters','Algorithms','Value'])  
df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar')  
plt.title("All Algorithms Performance Graph")  
plt.show()

#showing all algorithms with scenario A and B performance values  
columns = ["Algorithm Name","Precison","Recall","FScore","Accuracy"]  
values = []  
algorithm\_names = ["Existing Random Forest", "Propose CNN + BI-LSTM", "Extension CNN + BI-LSTM + BI-GRU"]  
for i in range(len(algorithm\_names)):  
 values.append([algorithm\_names[i],precision[i],recall[i],fscore[i],accuracy[i]])  
   
temp = pd.DataFrame(values,columns=columns)  
temp

Algorithm Name Precison Recall FScore \  
0 Existing Random Forest 27.399561 39.592668 31.015376   
1 Propose CNN + BI-LSTM 87.883684 80.660302 74.777627   
2 Extension CNN + BI-LSTM + BI-GRU 98.600411 98.794877 98.686285   
  
 Accuracy   
0 53.529044   
1 85.633979   
2 98.500937

#now predict pain type using test data  
testData = pd.read\_csv("Dataset/testData.csv")#reading test data  
testData.fillna(0, inplace = True)  
temp = testData.values  
testData = testData.values  
test = scaler.transform(testData)#normalizing values  
test = np.reshape(test, (test.shape[0], 34, 4))  
predict = extension\_model.predict(test)#performing prediction on test data using extension model object  
for i in range(len(predict)):  
 y\_pred = np.argmax(predict[i])  
 print("Test Data = "+str(temp[i])+" Predicted Pain Type ====> "+labels[y\_pred]+"\n")

Test Data = [ 28. 164. 130. 164. 144. 166. 157. 167. 169. 170. 182. 177. 194. 187.  
 203. 198. 210. 212. 212. 226. 210. 238. 205. 248. 197. 255. 187. 258.  
 175. 261. 163. 264. 151. 265. 138. 174. 120. 180. 114. 189. 112. 199.  
 113. 207. 117. 225. 117. 235. 115. 245. 116. 253. 119. 258. 126. 216.  
 129. 216. 138. 215. 147. 215. 157. 203. 163. 209. 165. 214. 167. 220.  
 166. 225. 164. 183. 131. 189. 128. 196. 128. 202. 132. 196. 133. 189.  
 133. 228. 133. 235. 131. 242. 132. 248. 135. 242. 136. 235. 136. 196.  
 182. 203. 177. 209. 175. 214. 177. 219. 175. 225. 178. 231. 183. 225.  
 188. 219. 189. 213. 190. 208. 189. 202. 186. 200. 182. 209. 181. 214.  
 181. 219. 181. 228. 183. 219. 181. 214. 181. 209.] Predicted Pain Type ====> Pain1  
  
Test Data = [ 0. 169. 132. 170. 145. 172. 158. 173. 171. 177. 184. 184. 195. 195.  
 204. 208. 211. 221. 213. 235. 210. 247. 204. 257. 195. 263. 184. 265.  
 172. 268. 160. 270. 148. 271. 135. 179. 122. 186. 116. 195. 113. 205.  
 113. 214. 117. 233. 117. 242. 115. 251. 115. 260. 118. 265. 124. 224.  
 129. 224. 139. 224. 148. 224. 158. 212. 164. 218. 166. 223. 168. 229.  
 166. 234. 165. 190. 132. 196. 129. 203. 128. 210. 132. 203. 133. 196.  
 134. 235. 133. 242. 130. 249. 130. 255. 134. 249. 136. 242. 135. 204.  
 181. 211. 178. 219. 176. 223. 177. 228. 176. 235. 178. 241. 181. 235.  
 187. 228. 190. 223. 191. 218. 190. 211. 187. 207. 181. 218. 181. 223.  
 182. 228. 181. 238. 181. 228. 181. 223. 182. 218.] Predicted Pain Type ====> Pain0  
  
Test Data = [ 1. 171. 129. 171. 142. 173. 155. 174. 167. 177. 180. 184. 191. 194.  
 200. 206. 207. 221. 210. 235. 208. 247. 203. 257. 194. 263. 184. 266.  
 172. 268. 160. 271. 148. 272. 136. 182. 118. 188. 111. 197. 108. 207.  
 109. 216. 113. 235. 114. 244. 112. 254. 113. 262. 116. 267. 124. 225.  
 126. 225. 135. 225. 144. 225. 153. 213. 160. 219. 162. 224. 163. 229.  
 163. 234. 161. 192. 128. 198. 125. 205. 125. 212. 129. 205. 130. 198.  
 131. 237. 131. 245. 128. 251. 128. 257. 133. 251. 134. 244. 133. 205.  
 178. 212. 174. 219. 172. 224. 173. 229. 172. 235. 175. 241. 180. 235.  
 185. 228. 187. 223. 187. 218. 186. 212. 184. 208. 178. 219. 177. 224.  
 178. 228. 178. 238. 179. 228. 178. 224. 179. 219.] Predicted Pain Type ====> Pain4  
  
Test Data = [ 6. 167. 131. 167. 144. 168. 156. 169. 169. 172. 181. 179. 193. 188.  
 202. 200. 209. 214. 212. 228. 211. 240. 205. 250. 198. 257. 187. 260.  
 175. 263. 163. 266. 151. 268. 138. 176. 120. 183. 114. 192. 112. 201.  
 112. 210. 115. 229. 117. 238. 115. 248. 116. 256. 120. 261. 127. 219.  
 128. 218. 138. 218. 147. 217. 156. 206. 163. 211. 165. 217. 166. 222.  
 165. 228. 164. 186. 131. 192. 128. 199. 128. 206. 131. 199. 132. 192.  
 133. 231. 133. 239. 130. 245. 131. 251. 135. 245. 136. 238. 135. 198.  
 180. 205. 176. 212. 174. 217. 176. 222. 175. 228. 177. 234. 182. 227.  
 186. 221. 188. 216. 189. 210. 188. 204. 185. 201. 180. 211. 180. 216.  
 180. 221. 180. 231. 181. 221. 180. 216. 181. 211.] Predicted Pain Type ====> Pain2  
  
Test Data = [ 6. 167. 130. 167. 143. 168. 156. 169. 169. 172. 181. 179. 193. 189.  
 202. 200. 209. 214. 212. 228. 210. 240. 205. 250. 197. 257. 187. 261.  
 175. 263. 163. 266. 151. 268. 138. 177. 119. 183. 114. 192. 112. 202.  
 112. 210. 116. 229. 117. 238. 115. 248. 116. 256. 120. 261. 127. 219.  
 129. 218. 138. 218. 147. 217. 156. 206. 163. 212. 164. 217. 166. 222.  
 165. 228. 164. 186. 131. 192. 128. 199. 128. 206. 131. 199. 133. 192.  
 133. 231. 134. 238. 130. 245. 131. 251. 135. 245. 137. 238. 136. 199.  
 180. 205. 176. 212. 174. 217. 176. 222. 175. 228. 177. 234. 182. 228.  
 187. 221. 189. 216. 189. 211. 188. 205. 185. 202. 180. 212. 180. 217.  
 180. 221. 180. 231. 182. 221. 180. 217. 181. 212.] Predicted Pain Type ====> Pain2  
  
Test Data = [ 24. 168. 132. 168. 145. 170. 158. 171. 171. 174. 183. 180. 195. 190.  
 204. 201. 212. 215. 215. 229. 213. 241. 208. 251. 201. 258. 191. 262.  
 179. 265. 167. 268. 155. 270. 142. 178. 123. 185. 118. 194. 116. 204.  
 117. 213. 121. 231. 122. 240. 120. 250. 121. 259. 125. 264. 132. 221.  
 133. 220. 143. 220. 153. 219. 163. 207. 168. 213. 169. 218. 171. 224.  
 170. 230. 169. 188. 133. 194. 131. 201. 131. 207. 135. 200. 136. 194.  
 136. 234. 137. 241. 135. 247. 136. 253. 139. 247. 141. 240. 139. 199.  
 184. 206. 181. 213. 179. 218. 181. 223. 180. 229. 182. 236. 186. 229.  
 191. 223. 193. 217. 193. 212. 192. 205. 189. 202. 184. 212. 184. 218.  
 185. 223. 185. 232. 186. 223. 185. 217. 185. 212.] Predicted Pain Type ====> Pain3  
  
Test Data = [ 24. 168. 131. 168. 145. 169. 158. 171. 171. 174. 183. 180. 195. 190.  
 205. 202. 212. 215. 215. 229. 213. 241. 208. 251. 200. 258. 190. 262.  
 179. 265. 167. 268. 154. 270. 142. 178. 123. 185. 118. 194. 116. 204.  
 117. 213. 121. 231. 122. 240. 121. 250. 121. 258. 125. 264. 132. 221.  
 133. 220. 143. 220. 153. 219. 163. 207. 168. 213. 170. 218. 172. 224.  
 171. 229. 169. 188. 133. 194. 131. 201. 131. 207. 135. 201. 136. 194.  
 136. 234. 137. 241. 135. 247. 136. 253. 139. 247. 141. 240. 139. 199.  
 184. 206. 181. 212. 179. 217. 181. 223. 180. 229. 182. 235. 186. 229.  
 191. 222. 193. 217. 193. 211. 192. 205. 189. 202. 184. 212. 184. 217.  
 185. 222. 185. 232. 186. 222. 185. 217. 185. 212.] Predicted Pain Type ====> Pain3  
  
Test Data = [ 28. 164. 130. 164. 144. 166. 157. 167. 169. 170. 182. 176. 194. 187.  
 203. 198. 210. 211. 212. 225. 211. 237. 206. 247. 199. 254. 189. 258.  
 177. 261. 164. 264. 152. 265. 139. 174. 120. 180. 114. 189. 112. 199.  
 113. 208. 117. 226. 117. 235. 116. 245. 116. 253. 119. 258. 126. 216.  
 129. 216. 138. 215. 147. 215. 157. 203. 163. 209. 165. 214. 167. 220.  
 166. 225. 164. 183. 131. 189. 128. 196. 128. 202. 132. 196. 133. 189.  
 133. 229. 134. 236. 131. 242. 132. 248. 135. 242. 136. 235. 136. 196.  
 182. 203. 177. 209. 175. 214. 176. 219. 175. 225. 178. 231. 183. 225.  
 187. 219. 189. 213. 189. 208. 189. 202. 186. 199. 181. 209. 180. 214.  
 181. 219. 181. 228. 183. 219. 181. 213. 181. 209.] Predicted Pain Type ====> Pain1  
  
Test Data = [ 0. 169. 132. 169. 146. 171. 159. 173. 172. 177. 184. 184. 196. 195.  
 204. 207. 211. 221. 213. 235. 211. 247. 205. 256. 196. 262. 185. 265.  
 173. 267. 161. 270. 148. 271. 136. 179. 122. 186. 116. 195. 113. 205.  
 113. 214. 117. 233. 117. 242. 115. 252. 115. 260. 118. 265. 124. 224.  
 129. 224. 138. 224. 148. 224. 158. 212. 164. 218. 166. 223. 168. 229.  
 166. 234. 165. 189. 132. 196. 129. 203. 129. 210. 132. 203. 133. 196.  
 134. 235. 133. 242. 129. 249. 130. 255. 134. 249. 135. 242. 135. 203.  
 181. 211. 178. 219. 176. 223. 177. 228. 176. 235. 178. 241. 181. 235.  
 187. 228. 190. 223. 190. 218. 190. 211. 187. 207. 181. 218. 181. 223.  
 182. 228. 181. 238. 181. 228. 181. 223. 182. 219.] Predicted Pain Type ====> Pain0  
  
Test Data = [ 1. 171. 129. 171. 142. 173. 155. 174. 167. 177. 180. 183. 192. 194.  
 201. 206. 207. 220. 210. 234. 208. 247. 203. 257. 195. 262. 184. 265.  
 172. 268. 161. 271. 149. 272. 136. 182. 118. 188. 111. 197. 108. 207.  
 109. 216. 113. 235. 114. 244. 112. 253. 113. 262. 116. 267. 124. 225.  
 126. 225. 135. 225. 144. 224. 153. 213. 160. 218. 162. 224. 163. 229.  
 163. 234. 161. 192. 128. 198. 125. 205. 125. 211. 129. 205. 130. 198.  
 130. 237. 131. 244. 127. 251. 128. 257. 132. 251. 134. 244. 133. 205.  
 178. 212. 174. 219. 172. 224. 173. 228. 172. 235. 175. 241. 180. 234.  
 184. 228. 186. 223. 187. 218. 186. 212. 183. 208. 178. 219. 177. 223.  
 178. 228. 178. 238. 179. 228. 178. 223. 179. 219.] Predicted Pain Type ====> Pain4

**6. TESTING**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

## **Implementation**

## The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## **Testing**

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### **System Testing**

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

**Integration Testing**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

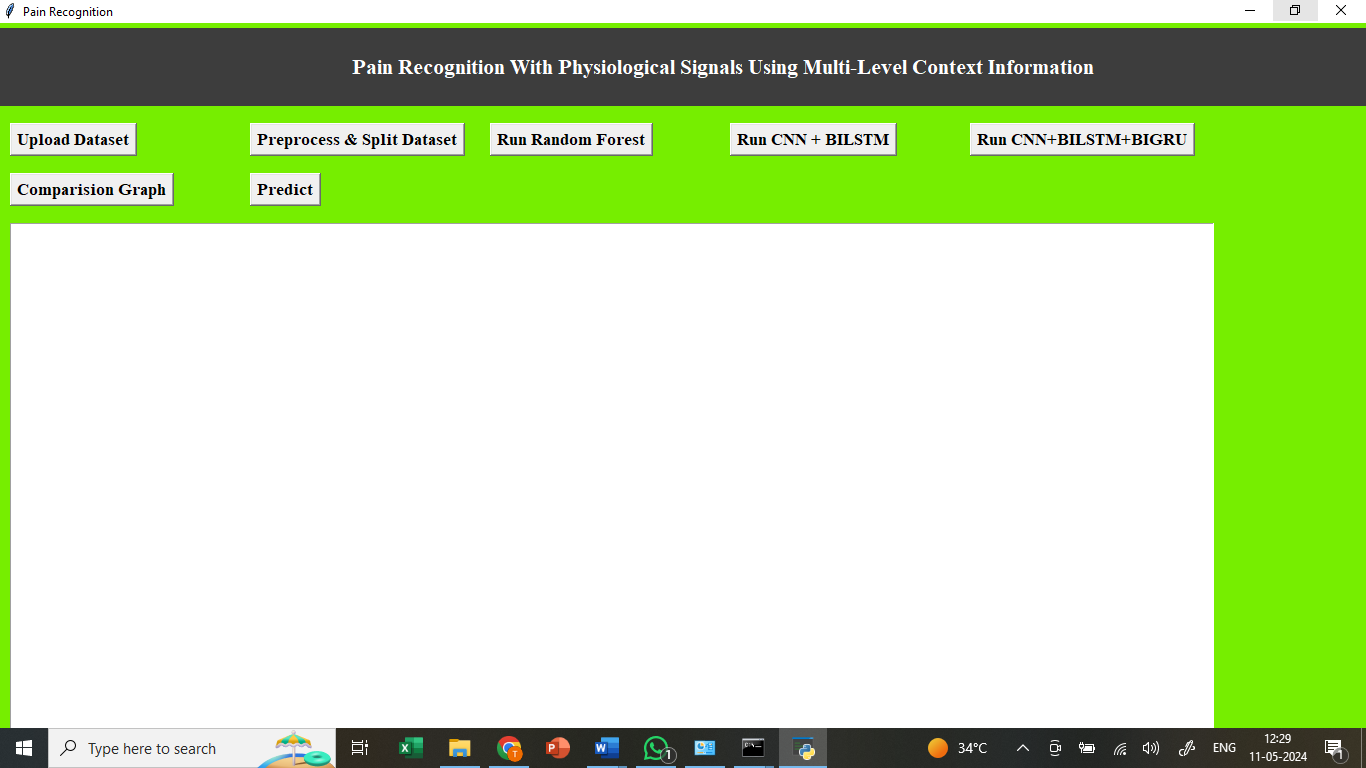
**Acceptance Testing**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation

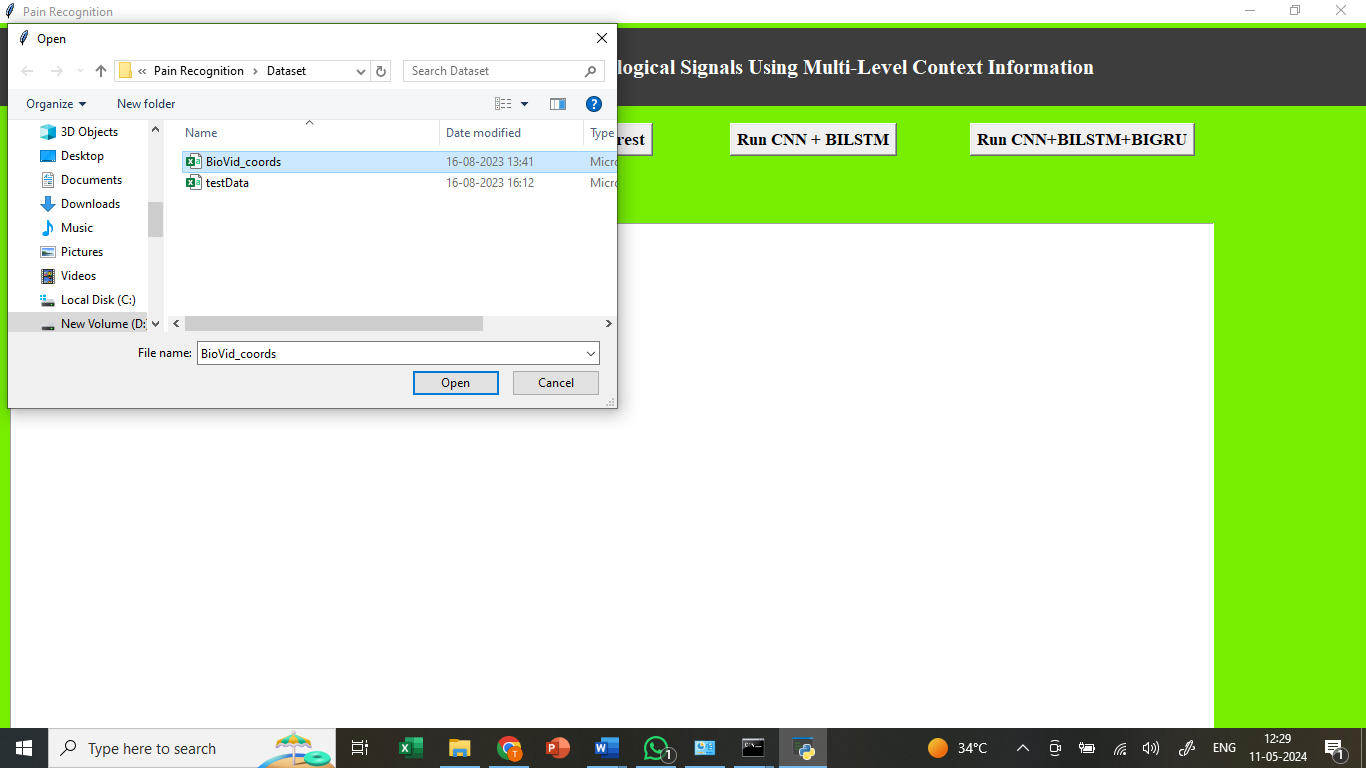
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | **Actual** |
| 01 | Upload Dataset | Test whether Dataset is uploaded or not into the system | If the Dataset may not uploaded | We cannot do further operations | Water Quality Dataset uploaded we will do further operations | High | High |
| 02 | Preprocess& Normalized Dataset | Test whether the Pre-process & Normalized Dataset Successfully or not | If the Pre-process & Normalized Dataset may not Run Successfully | We cannot do further operations | we will do further  operations | High | High |
| 03 | Run CalculateMetrices | Test whether CalculateMetrices Algorithm Run Successfully or not | If the  CalculateMetricesAlgorithm may not Run Successfully | We cannot do further operations | we will do further  operations | High | High |
| 04 | Train Run Random Forest Algorithm | Test whether Random Forest Algorithm Run Successfully or not | If the  Random Forest Algorithm may not Run Successfully | We cannot do further operations | we will do further  operations | High | High |
| 05 | Predict | Test prediction whether Successfully or not | If the  Predict may not Successfully | We cannot do further operations | we will do further  operations | High | High |

**7. SCREENSHOTS**:

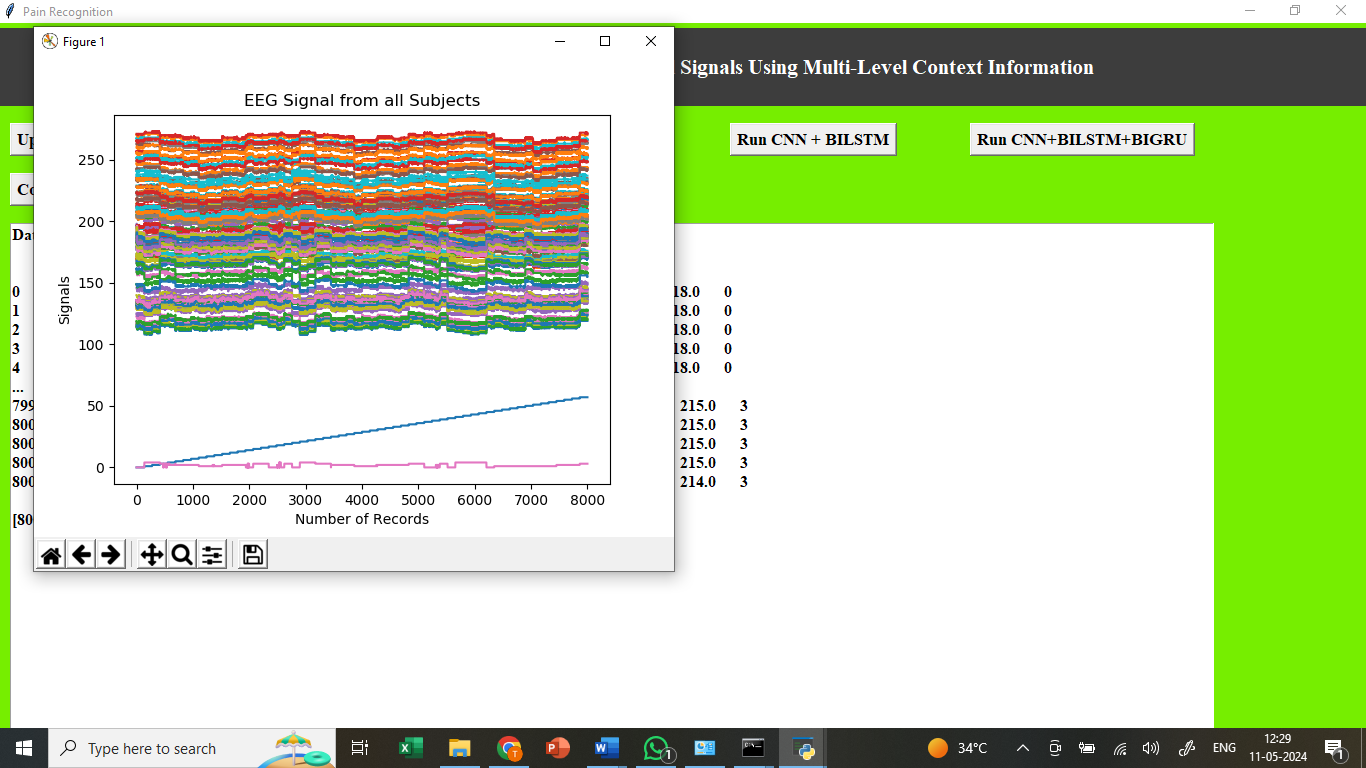
To Run the application Click on “run.bat” file from the file location.



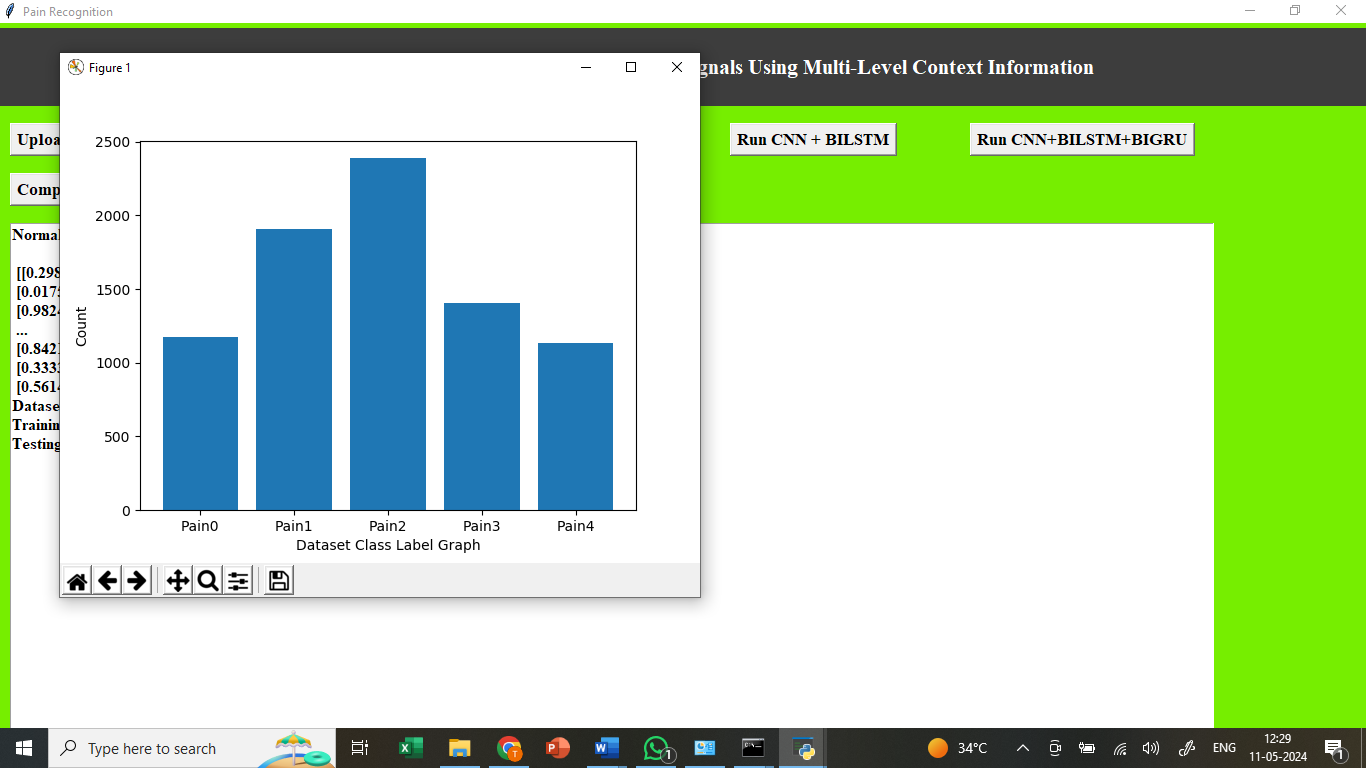
In the above screen we got Tkinter Output Window. Now Click On the “Upload Dataset” button to upload the dataset to the application.



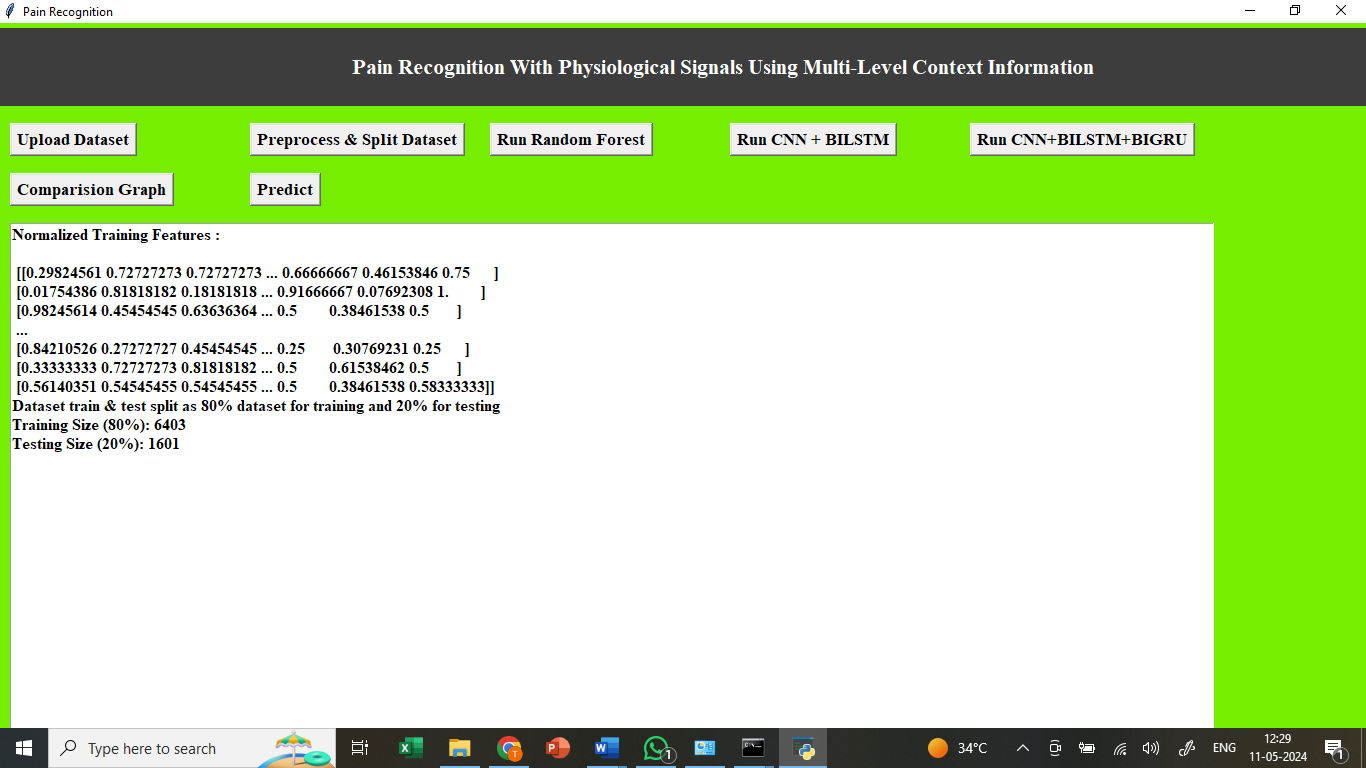
In the above screen we can see the dataset uploading.



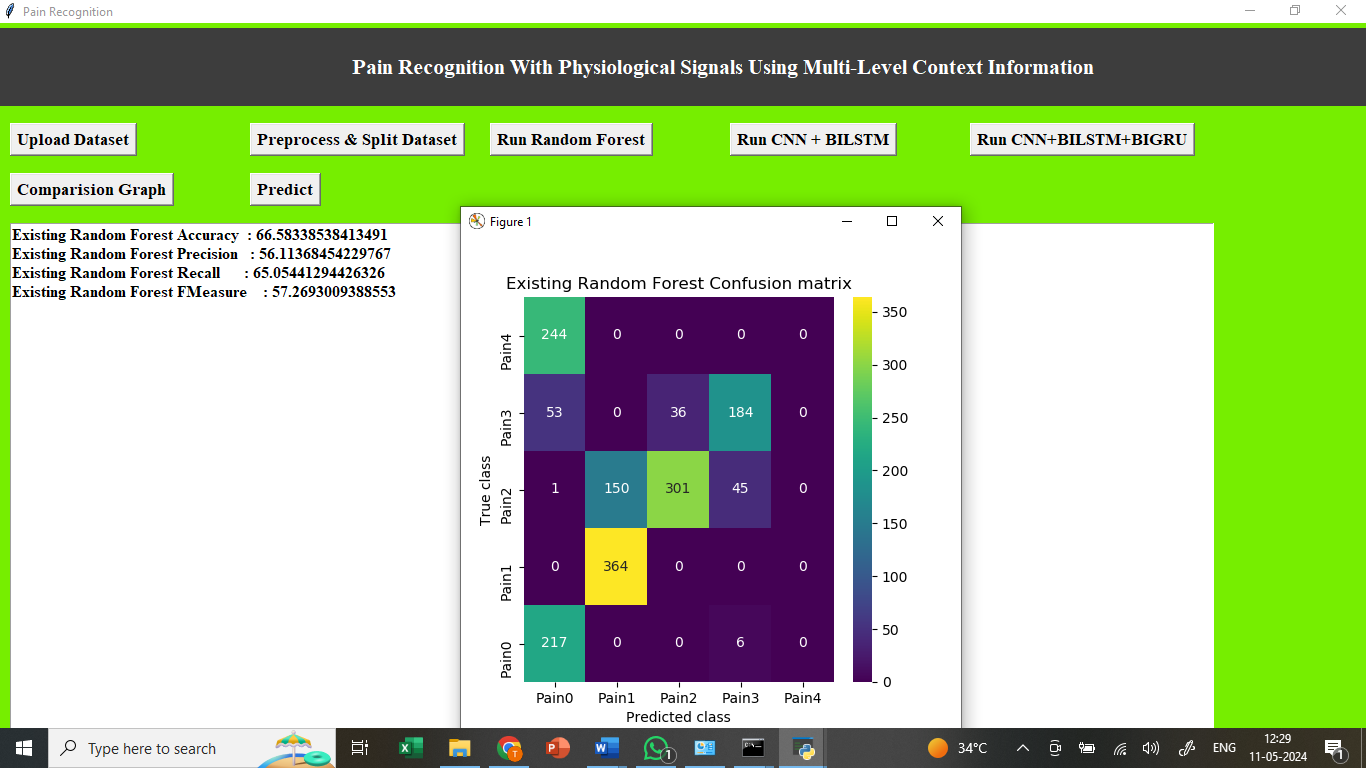
In above screen defining code for features normalization and then reading and displaying dataset where first column contains subject ID and last column contains Pains labels from 0 to 4. In above graph plotting EEG signals from all subjects where x-axis represents EEG record number and y-axis represents EEG signal values from all subjects.



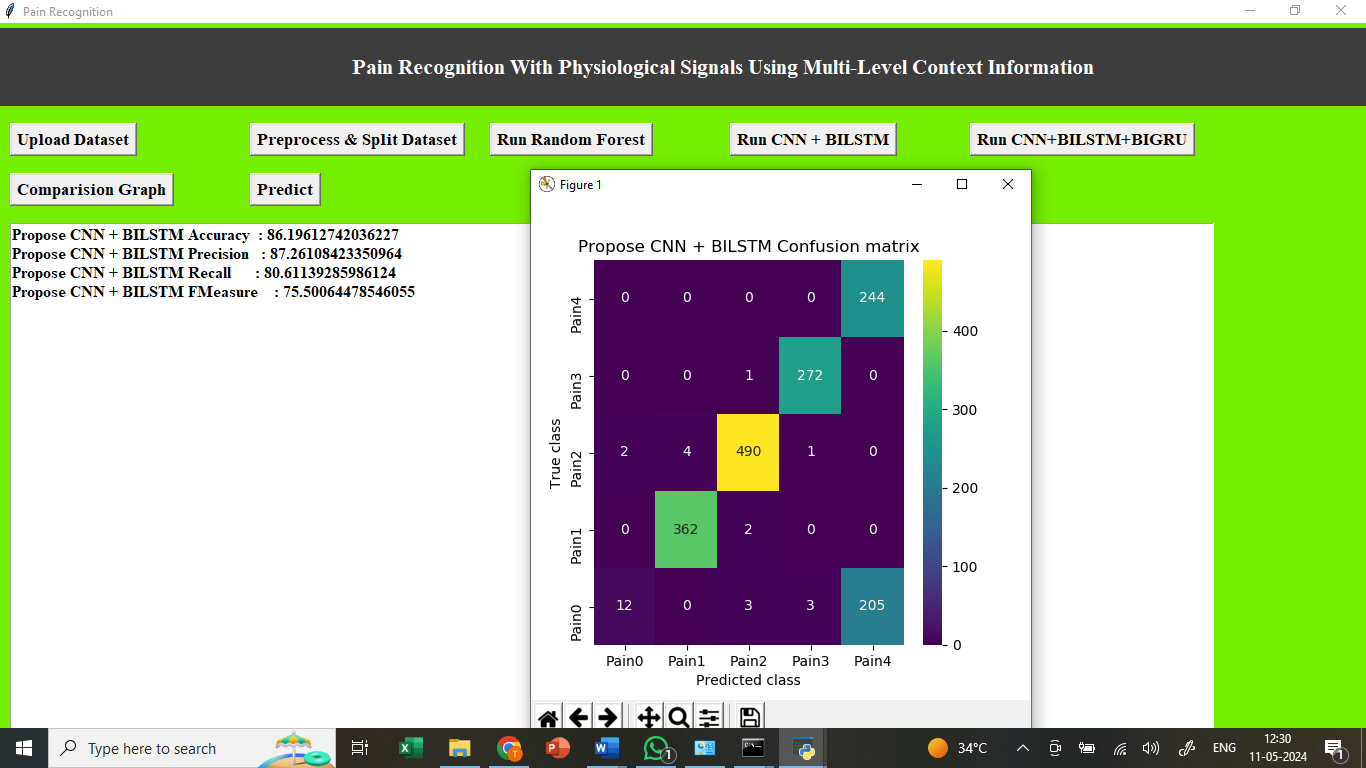
In above screen finding and plotting graphs of different pains and count of those pains ECG signals and in graph x-axis represents Pain Type and y-axis represents count.



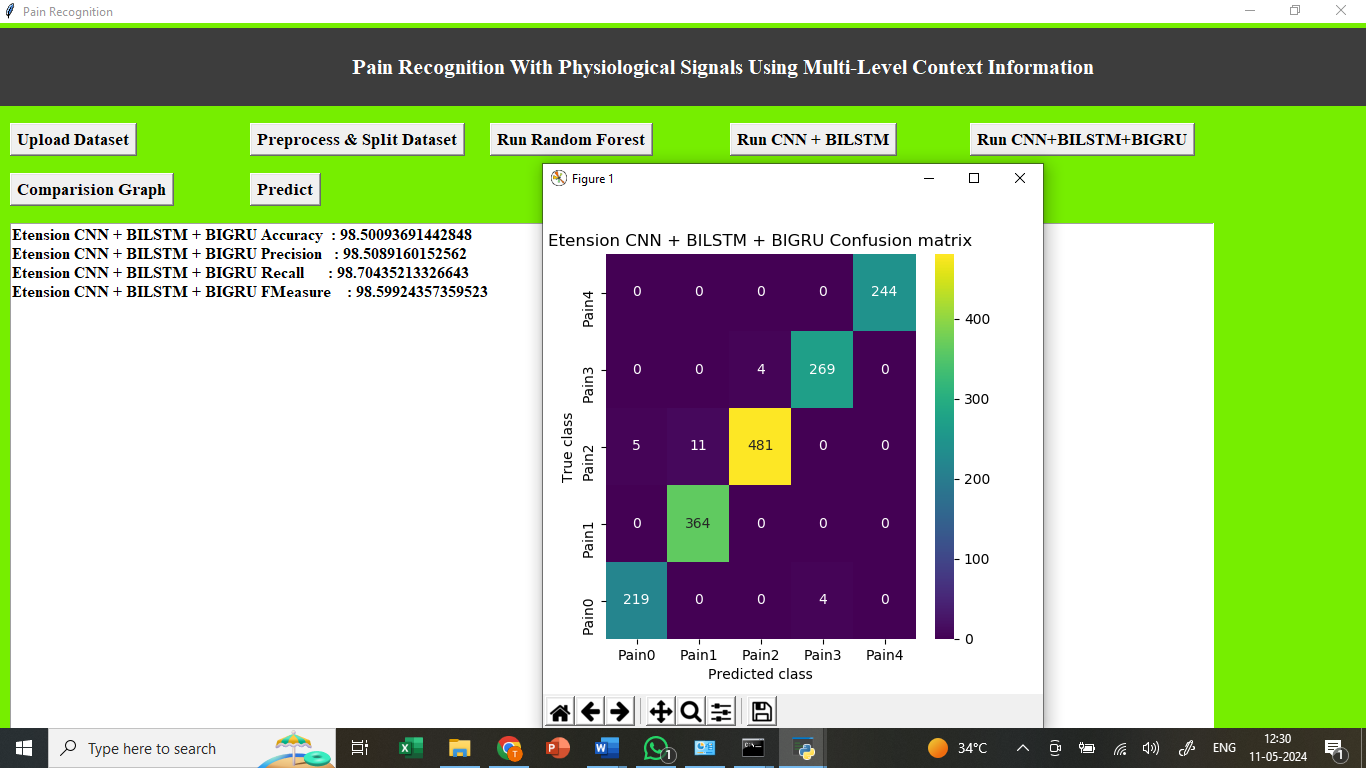
In above screen performing various dataset processing like features normalization, shuffling and splitting into train and test. Now click on “Run Random Forest Algorithm”.



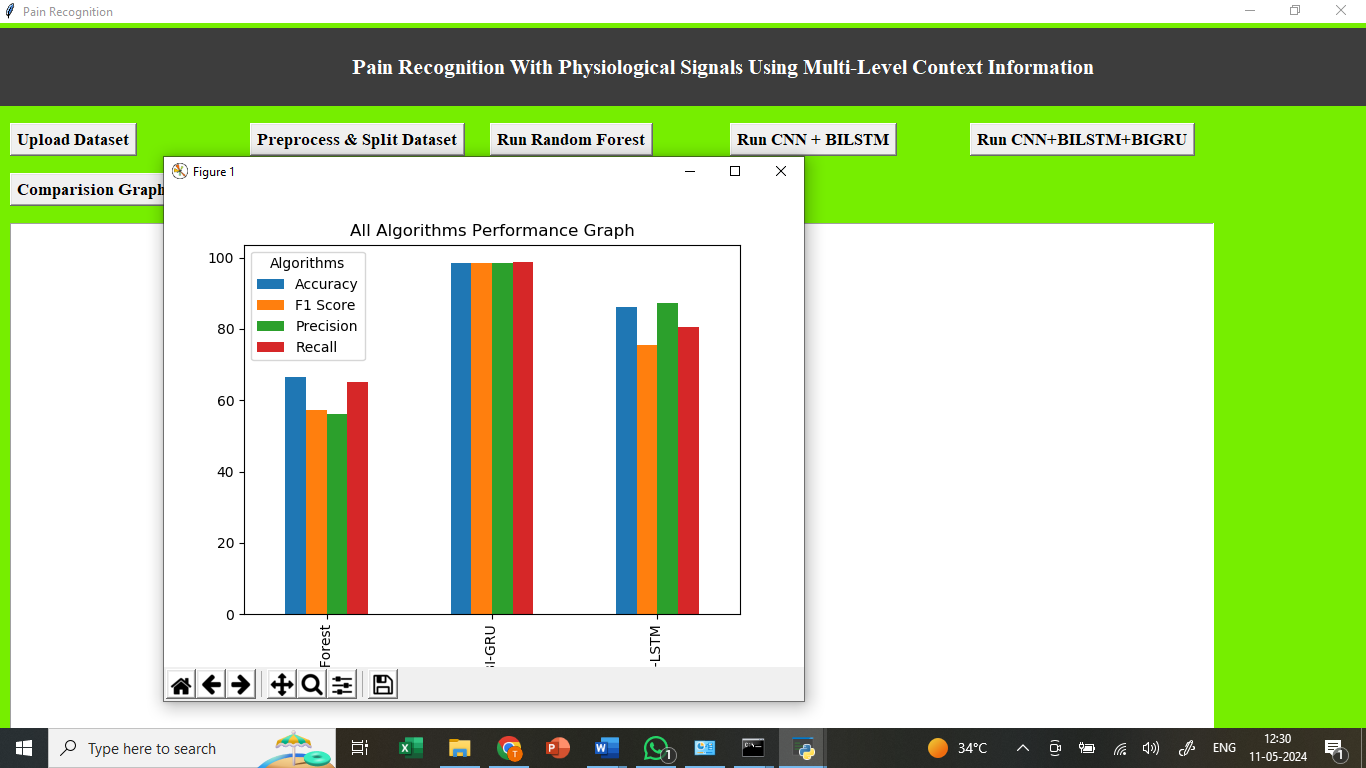
In above screen training existing algorithm called Random Forest and after execution will get output. In above screen Random Forest got 53% accuracy and LOSO also as 53% and in confusion matrix graph x-axis represents Predicted Pain Labels and y-axis represents True Labels and we can see Random Forest predict all test data into single Pain Type which is wrong and below is the propose algorithm.



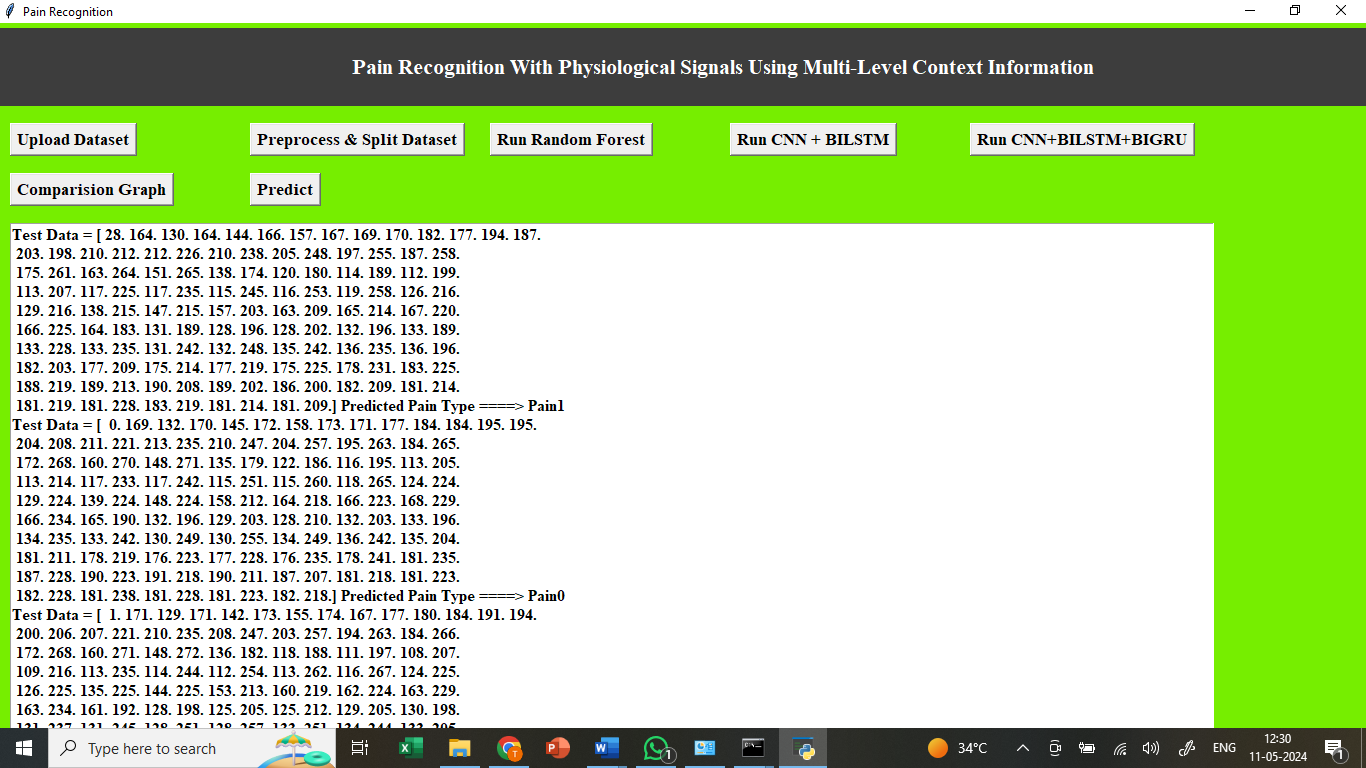
In above propose algorithm we are defining multilevel algorithm with CNN + BILSTM and after execution will get output In above screen with propose multilevel algorithm we got 85% accuracy and in confusion matrix graph x-axis represents Predicted Labels and y-axis represents True Labels and all different colour boxes in diagnol contains correct prediction count and remaining blue boxes contains incorrect prediction count and in above graph we can see propose algorithm predicted all types of PAINS.



In above screen defining extension algorithm with CNN + BILSTM + BIGRU with 3 multilevel algorithms for features selection and optimizations and after execution of above block will get output In above screen extension got 98% accuracy and in confusion matrix we can see all pains are predicted correctly which you can analyse in diagnol both x and y-axis pains are matching.



In above graph x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars and in all algorithms Extension got high accuracy.



Above code reading test data from file and then normalizing and predicting pain type using extension model and in output before =🡺 symbol we can see ECG test data and after =🡺 arrow symbol we can see predicted PAIN type. In above screen predicted pain type showing in blue colour

**8. CONCLUSION:**

This paper proposes a deep learning approach based on physiological signals for pain recognition. Our method has the role of feature extraction and classification, completely replacing manual extraction methods that require highly specialized knowledge. We propose multi-level context information explored from hidden sequence information. Specifically, the architecture employs hidden information for the attention mechanism to create the context vector. We combine hidden information and context vector to create the context information. Combining context information at three levels produces multi-level context information. We perform binary classification between baseline and different pain intensities based on Part A of the BioVid Heat Pain database. In addition, we also perform binary classification based on the Emopain 2021 dataset. Our experimental results prove that multi-level context information has more significance than uni-level context information based on Part A of the BioVid Heat Pain database and the Emopain 2021 dataset. Our results demonstrate the great significance of EDA in pain classification. Combining EDA and ECG mostly provides good performance in classification tasks based on Part A of the BioVid Heat Pain database.

**9. REFERENCES:**

[1] P. Werner, D. Lopez-Martinez, S. Walter, A. Al-Hamadi, S. Gruss, and R. W. Picard, ‘‘Automatic recognition methods supporting pain assessment: A survey,’’ IEEE Trans. Affect. Comput., vol. 13, no. 1, pp. 530–552, Jan. 2022.

[2] S. Walter, S. Gruss, H. Ehleiter, J. Tan, H. C. Traue, S. Crawcour, P. Werner, A. Al-Hamadi, and A. O. Andrade, ‘‘The biovid heat pain database data for the advancement and systematic validation of an automated pain recognition system,’’ in Proc. IEEE Int. Conf. Cybern. (CYBCO), Jun. 2013, pp. 128–131.

[3] Y. LeCun, Y. Bengio, and G. Hinton, ‘‘Deep learning,’’ Nature, vol. 521, no. 7553, pp. 436–444, Sep. 2015.

[4] T. Olugbade, R. Sagoleo, S. Ghisio, N. Gold, C. D. C. Amanda, B. D. Gelder, A. Camurri, G. Volpe, and N. Bianchi-Berthouze, ‘‘The affectmove 2021 challenge—Affect recognition from naturalistic movement data,’’ in Proc. 9th Int. Conf. Affect. Comput. Intell. Interact. Workshops Demos (ACIIW), Sep./Oct. 2021, pp. 1–5.

[5] P. Werner, A. Al-Hamadi, and S. Walter, ‘‘Analysis of facial expressiveness during experimentally induced heat pain,’’ in Proc. 7th Int. Conf. Affect. Comput. Intell. Interact. Workshops Demos (ACIIW), Oct. 2017, pp. 176–180.

[6] P. Werner, A. Al-Hamadi, R. Niese, S. Walter, S. Gruss, and H. C. Traue, ‘‘Automatic pain recognition from video and biomedical signals,’’ in Proc. 22nd Int. Conf. Pattern Recognit., Aug. 2014, pp. 4582–4587.

[7] L. Breiman, ‘‘Random forests,’’ Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.

[8] M. Kachele, P. Thiam, M. Amirian, F. Schwenker, and G. Palm, ‘‘Methods for person-centered continuous pain intensity assessment from biophysiological channels,’’ IEEE J. Sel. Topics Signal Process., vol. 10, no. 5, pp. 854–864, Aug. 2016.

[9] Y. Chu, X. Zhao, J. Han, and Y. Su, ‘‘Physiological signal-based method for measurement of pain intensity,’’ Front. Neurosci., vol. 11, p. 279, May 2017.

[10] R. A. Fisher, ‘‘The use of multiple measurements in taxonomic problems,’’ Ann. Eugenics, vol. 7, no. 2, pp. 179–188, 1936.

[11] S. Abe, Support Vector Machines for Pattern Classification, vol. 2. London, U.K.: Springer, 2005.

[12] F. Pouromran, S. Radhakrishnan, and S. Kamarthi, ‘‘Exploration of physiological sensors, features, and machine learning models for pain intensity estimation,’’ PLoS ONE, vol. 16, no. 7, Jul. 2021, Art. no. e0254108.

[13] A. Smola and B. Schölkopf, ‘‘A tutorial on support vector regression,’’ Statist. Comput., vol. 14, no. 3, pp. 199–222, 2014.

[14] T. Chen and C. Guestrin, ‘‘XGBoost: A scalable tree boosting system,’’ in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2016, pp. 785–794.

[15] F. Pouromran, Y. Lin, and S. Kamarthi, ‘‘Personalized deep bi-LSTM RNN based model for pain intensity classification using EDA signal,’’ Sensors, vol. 22, no. 21, p. 8087, Oct. 2022.