**Explainable Artificial Intelligence for Patient Safety: A Review of Application in Pharmacovigilance**

**1. INTRODUCTION:**

The World Health Organization defines pharmacovigilance (PV) as the science and activities related to the detection, assessment, understanding, and prevention of adverse effects or other drug-related problems. Recent artificial intelligence-based technologies can be an efficient complement to traditional PV methods, which can be costly and time-consuming and can result in adverse drug reactions (ADRs) that go unreported to healthcare professionals. Artificial intelligence (AI) can improve PV, but its use in PV is still in the early stages of research. Various machine learning (ML) techniques, together with natural language The associate editor coordinating the review of this manuscript and approving it for publication was Jerry Chun-Wei Lin. processing and data mining, can be applied to electronic health records, claims databases and social media data to improve the characterization of known drug side effects and reactions, and to detect new signals. AI-based technologies have been criticized for their inexplicable algorithms, despite their high predictive power. In critical decision areas such as healthcare, the reasoning behind a decision is as important as the decision itself, which is why there is growing interest in and research and development around Explainable Artificial Intelligence (XAI). XAI was developed to improve the transparency of AI systems and generate explanations for them, and seeks to increase trust and understanding by assessing the strengths and limitations of existing models. Approaches that extract information from a model’s decision-making process, such as post-hoc explanations, can provide useful 50830 This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/ VOLUME 11, 2023 S. Lee et al.: XAI for Patient Safety: A Review of Application in Pharmacovigilance information for practitioners and users interested in caseby-case explanations rather than the internal workings of a model. XAI increases the explainability and transparency of AI algorithms by making it possible to interpret the variables that influence decisions, complex internal features, and learned decision paths within a decision process. I.R. Ward et al. successfully quantified the importance of features using an XAI algorithm, further demonstrating the potential contribution of XAI to PV monitoring. The importance of PV in medicine is relevant to all species affected by medical interventions, and ensuring medical safety requires attention and research into approaches such as drug safety reporting and the exchange of reliable and timely information on PV activities. The global pharmacovigilance and drug safety software market size was valued at USD 6.9 billion in 2021 and is estimated to expand at a compound annual growth rate (CAGR) of 10.5% between 2022 and 2030 (Source: www.grandviewresearch.com). The aim of this study was to review the literature on the use of XAI in PV by identifying publications related to ML/AI and drugs and the rationale for the reported findings. From the perspective of AI and XAI usage, these studies were analyzed, and the findings were summarized, in which the use of XAI in the field of PV is referred to as ‘‘PV XAI’’. The main contributions are highlighted and discussed below: - This study is clearly an early attempt to review XAI research in PV. Unlike other fields, we found that XAI research in PV is at an early stage of development, limited to a few articles and some methodologies. - Nevertheless, we have identified the positive potential of PV XAI for drug therapy, ADRs, polypharmacy and drug repurposing. - While safety issues in real-world healthcare settings may limit the growth of the field, we expect PV XAI research to expand as it has in other areas, and we encourage collaboration and ongoing research discussions with experts in the field.

* 1. **Objective of the project:**

Explainable AI (XAI) is a methodology that complements the black box of artificial intelligence, and its necessity has recently been highlighted in various fields. The purpose of this research is to identify studies in the field of pharmacovigilance using XAI. Though there have been many previous attempts to select papers, with a total of 781 papers being confirmed, only 25 of them manually met the selection criteria. This study presents an intuitive review of the potential of XAI technologies in the field of pharmacovigilance. In the included studies, clinical data, registry data, and knowledge data were used to investigate drug treatment, side effects, and interaction studies based on tree models, neural network models, and graph models. Finally, key challenges for several research issues for the use of XAI in pharmacovigilance were identified. Although artificial intelligence (AI) is actively used in drug surveillance and patient safety, gathering adverse drug reaction information, extracting drug-drug interactions, and predicting effects, XAI is not normally utilized. Therefore, the potential challenges involved in its use alongside future prospects should be continuously discussed.

**2. LITERATURE SURVEY:**

**Artificial intelligence in pharmacovigilance: An introduction to terms, concepts, applications, and limitations**

The tools of artificial intelligence (AI) have enormous potential to enhance activities in pharmacovigilance. Pharmacovigilance experts need not be AI experts, but they should know enough about AI to explore the possibilities of collaboration with those who are. Modern concepts of AI date from Alan Turing's work, especially his paper on "the imitation game", in the late 1940s and early 1950s. Its scope today includes computational skills, including the formulation of mathematical proofs; visual perception, including facial recognition and virtual reality; decision making by expert systems; aspects of language, such as language processing, speech recognition, creative composition, and translation; and combinations of these, e.g. in self-driving vehicles. Machines can be programmed with the ability to learn, using neural networks that mimic cognitive actions of the human brain, leading to deep structural learning. Limitations of AI include difficulties with language, arising from the need to understand context and interpret ambiguities, which particularly affect translation, and inadequacies of databases, requiring careful preparation and curation. New techniques may cause unforeseen difficulties via unexpected malfunctioning. Relevant terms and concepts include different types of machine learning, neural networks, natural language programming, ontologies, and expert systems. Adoption of the tools of AI in pharmacovigilance has been slow. Machine learning, in conjunction with natural language processing and data mining, to study adverse drug reactions in databases such as those found in electronic health records, claims databases, and social media, has the potential to enhance the characterization of known adverse effects and reactions and detect new signals.

**Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models**

With the availability of large databases and recent improvements in deep learning methodology, the performance of AI systems is reaching or even exceeding the human level on an increasing number of complex tasks. Impressive examples of this development can be found in domains such as image classification, sentiment analysis, speech understanding or strategic game playing. However, because of their nested non-linear structure, these highly successful machine learning and artificial intelligence models are usually applied in a black box manner, i.e., no information is provided about what exactly makes them arrive at their predictions. Since this lack of transparency can be a major drawback, e.g., in medical applications, the development of methods for visualizing, explaining and interpreting deep learning models has recently attracted increasing attention. This paper summarizes recent developments in this field and makes a plea for more interpretability in artificial intelligence. Furthermore, it presents two approaches to explaining predictions of deep learning models, one method which computes the sensitivity of the prediction with respect to changes in the input and one approach which meaningfully decomposes the decision in terms of the input variables.

**Transparency and trust in human-AI-interaction: The role of model-agnostic explanations in computer vision-based decision support**

Computer Vision, and hence Artificial Intelligence-based extraction of information from images, has increasingly received attention over the last years, for instance in medical diagnostics. While the algorithms' complexity is a reason for their increased performance, it also leads to the "black box" problem, consequently decreasing trust towards AI. In this regard, "Explainable Artificial Intelligence" (XAI) allows to open that black box and to improve the degree of AI transparency. In this paper, we first discuss the theoretical impact of explainability on trust towards AI, followed by showcasing how the usage of XAI in a health-related setting can look like. More specifically, we show how XAI can be applied to understand why Computer Vision, based on deep learning, did or did not detect a disease (malaria) on image data (thin blood smear slide images). Furthermore, we investigate, how XAI can be used to compare the detection strategy of two different deep learning models often used for Computer Vision: Convolutional Neural Network and Multi-Layer Perceptron. Our empirical results show that i) the AI sometimes used questionable or irrelevant data features of an image to detect malaria (even if correctly predicted), and ii) that there may be significant discrepancies in how different deep learning models explain the same prediction. Our theoretical discussion highlights that XAI can support trust in Computer Vision systems, and AI systems in general, especially through an increased understandability and predictability.

**Opportunities and challenges in explainable artificial intelligence (XAI): A survey**

Nowadays, deep neural networks are widely used in mission critical systems such as healthcare, self-driving vehicles, and military which have direct impact on human lives. However, the black-box nature of deep neural networks challenges its use in mission critical applications, raising ethical and judicial concerns inducing lack of trust. Explainable Artificial Intelligence (XAI) is a field of Artificial Intelligence (AI) that promotes a set of tools, techniques, and algorithms that can generate high-quality interpretable, intuitive, human-understandable explanations of AI decisions. In addition to providing a holistic view of the current XAI landscape in deep learning, this paper provides mathematical summaries of seminal work. We start by proposing a taxonomy and categorizing the XAI techniques based on their scope of explanations, methodology behind the algorithms, and explanation level or usage which helps build trustworthy, interpretable, and self-explanatory deep learning models. We then describe the main principles used in XAI research and present the historical timeline for landmark studies in XAI from 2007 to 2020. After explaining each category of algorithms and approaches in detail, we then evaluate the explanation maps generated by eight XAI algorithms on image data, discuss the limitations of this approach, and provide potential future directions to improve XAI evaluation.

**Explainable artificial intelligence for pharmacovigilance: What features are important when predicting adverse outcomes?**

Explainable Artificial Intelligence (XAI) has been identified as a viable method for determining the importance of features when making predictions using Machine Learning (ML) models. In this study, we created models that take an individual's health information (e.g. their drug history and comorbidities) as inputs, and predict the probability that the individual will have an Acute Coronary Syndrome (ACS) adverse outcome.

**Attention-based multi-task learning in pharmacovigilance**

Pharmacovigilance (PV) is the process of monitoring and assessing adverse events (AEs). Currently, the primary method of analysis in PV is manual inspection by individual case managers. However, this is largely unsustainable due to the increased volume of cases over the past few years. Since AE processing involves several sub-tasks, such as annotation and classification, our paper explores a novel solution to PV. In this paper, we propose a multi-task learning (MTL) model, where the hidden layers are shared, to jointly learn the tasks (Named Entity Recognition (NER), Classification). The results of our paper demonstrate that MTL is able to outperform our baseline classification model and equal the baseline model for annotation/NER. 11 Disclaimer: This content is for general information purposes only, and should not be used as a substitute for consultation with professional advisors.

**3. SYSTEM ANALYSIS**

**3.1 Existing System**

Machine and deep learning algorithms are using everywhere to predict future values or to classify objects but still some organizations are hesitating to employee this models for their business growth prediction. In such organization Medical Health is the one which is hesitating to employ machine learning models to predict drug side effects or their effectiveness

**Disadvantages:**

1. Less Accuracy
2. More time taking process

**3.2 PROPOSED SYSTEM**

In proposed system, using XGBOOT algorithm, GRAPHCNN, MLP and CNN2D algorithm which will optimized or filtered features using 2 dimension neurons. Optimized features help in getting enhance accuracy.

**Advantages:**

1. High Accuracy
2. Takes less time

**Modules Information:**

To implement this project we have used same dataset given in your requirement file and to implement this project we have designed following modules

1. Upload dataset: using this module, we will upload datasets
2. Dataset preprocessing: using this module, we will split and clean the data
3. Run Xgboost Algorithm: using this module, training XGBOOST algorithm as tree model and it got 95% accuracy
4. Run GRAPHCNN Algorithm: using this module, training GRAPHCNN algorithm and got 93% accuracy
5. Run Neural Network Algorithm: using this module, training Neural Network Algorithm and got 91% accuracy
6. Run Extension CNN2d Algorithm: using this module, training extension CNN2d algorithm and got 98% accuracy
7. Comparison Graph: using this module, displaying comparison between all algorithms where x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars and in all algorithms extension CNN2D got high performance
8. Upload Test data: using this module, Test data is uploaded for predicting drug

**FUNCTIONAL REQUIREMENTS:**

**SOFTWARE REQIREMENTS:**

**System Atributes:**

1. Filename
2. dataset
3. X, Y, mse, X\_train, X\_test, y\_train, y\_test

**Prototype:**

python 3.7.0 or 3.7.4

opencv-python==4.5.1.48

keras==2.3.1

tensorflow==1.14.0

protobuf==3.16.0

h5py==2.10.0

sklearn-extensions==0.0.2

scikit-learn==0.22.2.post1

Numpy

Pandas

**NON-FUNCTIONAL REQUIREMENT:**

**Usability:**  Usability is a quality attribute that assesses how easy user interfaces are to use. The word "usability" also refers to methods for improving ease-of-use during the design process.(how it was handle entire project easy)

**Security:** the quality or state of being secure: such as. a : freedom from danger : safety. b : freedom from fear or anxiety. c : freedom from the prospect of being laid off job security.

**Readability:** Readability is the ease with which a reader can understand a written text.

**Performance**: the execution of an action. : something accomplished : deed, feat. : the fulfillment of a claim, promise, or request : implementation. 3. : the action of representing a character in a play.

**Availability**: the quality or state of being available trying to improve the availability of affordable housing. 2 : an available person or thing.

**Scalability**: Scalability is the measure of a system's ability to increase or decrease in performance and cost in response to changes in application and system processing demands.

**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 artifacts will be produced. Software artifacts 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 artifacts, 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 artifact is linked to a specific design element, and that each developed artifact 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 artifacts, 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 artifacts, 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 labor 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 behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Nonfunctional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development lifecycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) 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](http://en.wikipedia.org/wiki/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 economical 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 economical 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.

**SYSTEM REQUIREMENT:**

**HARDWARE REQUIREMENTS:**

# Processor - Intel i3(min)

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

**SOFTWARE REQUIREMENTS:**

* Operating System - Windows10(min)
* Programming Language - Python with Jupiter notebook

**4. SYSTEM DESIGN**

**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



**USECASE 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 we



**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 compo



**DEPLOYMENT DIAGRAM:**

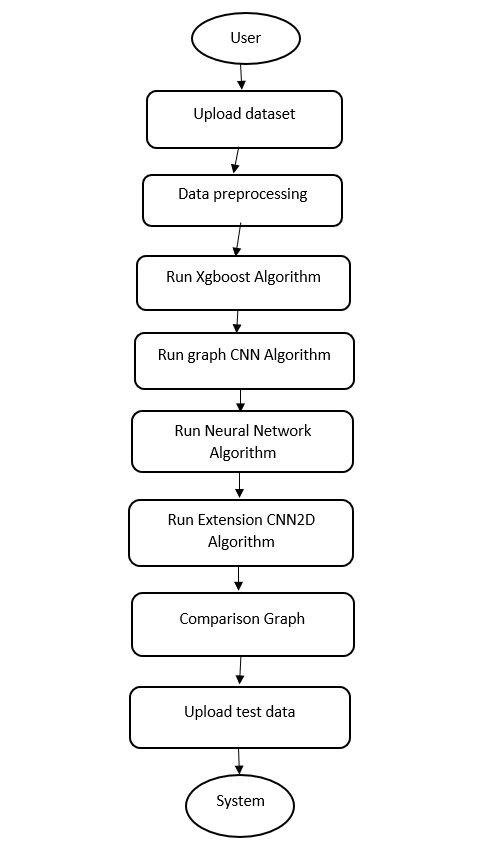
A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software 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:**

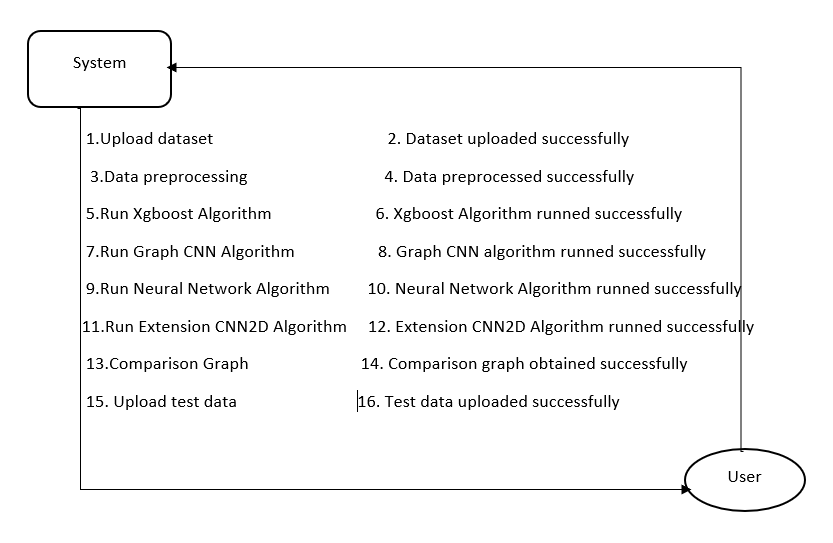
Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent



**Data flow:**

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.



**5. IMPLEMETATION**

**5.1 Python**

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

**History of Python:**

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

**Why Python was created?**

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

**Why the name Python?**

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

**Features of Python:**

**A simple language which is easier to learn**

Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax.

If you are a newbie, it's a great choice to start your journey with Python.

**Free and open-source**

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software’s written in it, you can even make changes to the Python's source code.

Python has a large community constantly improving it in each iteration.

**Portability**

You can move Python programs from one platform to another, and run it without any changes.

It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

**Extensible and Embeddable**

Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.

This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

**A high-level, interpreted language**

Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.

Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

**Large standard libraries to solve common tasks**

Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQL dB library using import MySQL db.

Standard libraries in Python are well tested and used by hundreds of people. So, you can be sure that it won't break your application.

**Object-oriented**

Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.

With OOP, you are able to divide these complex problems into smaller sets by creating objects.

**Applications of Python:**

**1. Simple Elegant Syntax**

Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

a = 2

b = 3

sum = a + b

print(sum)

**2. Not overly strict**

You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.

Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

**3. Expressiveness of the language**

Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

**4. Great Community and Support**

Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck.

**XGBOOST Algorithm:**

**XGBoost** is an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning models. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction. XGBoost stands for “Extreme Gradient Boosting” and it has become one of the most popular and widely used machine learning algorithms due to its ability to handle large datasets and its ability to achieve state-of-the-art performance in many machine learning tasks such as classification and regression.

One of the key features of XGBoost is its efficient handling of missing values, which allows it to handle real-world data with missing values without requiring significant pre-processing. Additionally, XGBoost has built-in support for parallel processing, making it possible to train models on large datasets in a reasonable amount of time.

XGBoost can be used in a variety of applications, including Kaggle competitions, recommendation systems, and click-through rate prediction, among others. It is also highly customizable and allows for fine-tuning of various model parameters to optimize performance.

XgBoost stands for Extreme Gradient Boosting, which was proposed by the researchers at the University of Washington. It is a library written in C++ which optimizes the training for Gradient Boosting.

**Before understanding the XGBoost, we first need to understand the trees especially the decision tree:**

### **Decision Tree:**

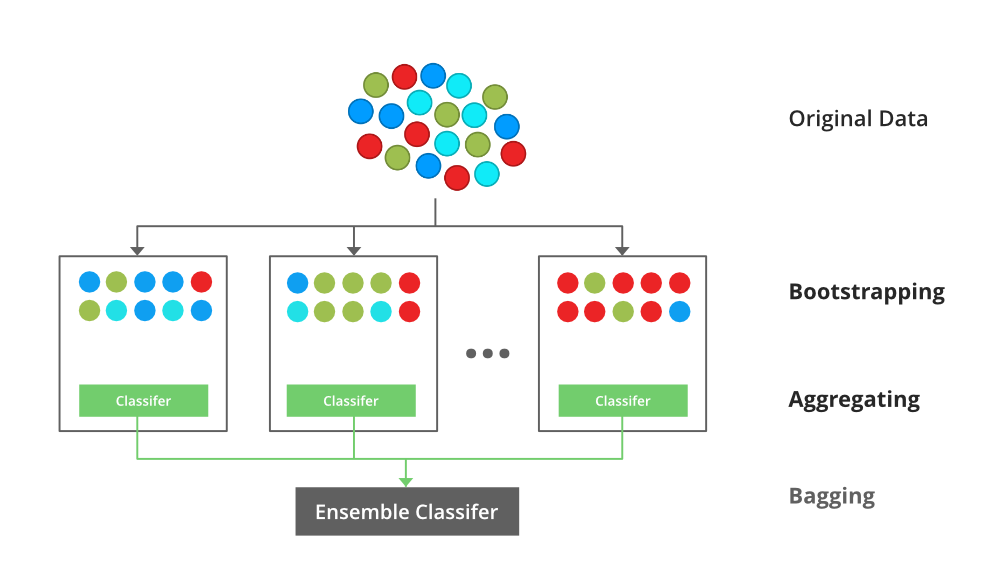
A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

A tree can be “learned” by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions.

### **Bagging**:

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.  
Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N examples(or data) from the original training dataset, where N is the size of the original training set. The training set for each of the base classifiers is independent of each other. Many of the original data may be repeated in the resulting training set while others may be left out.

Bagging reduces overfitting (variance) by averaging or voting, however, this leads to an increase in bias, which is compensated by the reduction in variance though.



### **Random Forest**:

Every decision tree has high variance, but when we combine all of them together in parallel then the resultant variance is low as each decision tree gets perfectly trained on that particular sample data and hence the output doesn’t depend on one decision tree but multiple decision trees. In the case of a classification problem, the final output is taken by using the majority voting classifier. In the case of a regression problem, the final output is the mean of all the outputs. This part is Aggregation.

The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.rRandom Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called Bootstrap.

### **Boosting**:

Boosting is an ensemble modelling, technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.



### **Gradient Boosting**

Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor’s error. In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels.

There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees).

### **XGBoost**

XGBoost is an implementation of Gradient Boosted decision trees. XGBoost models majorly dominate in many Kaggle Competitions.

In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

**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.

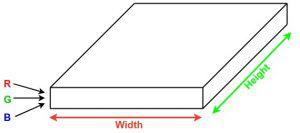


*Simple CNN architecture*

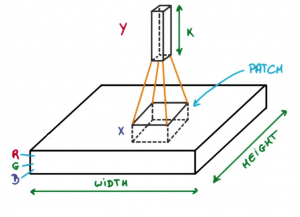
The Convolutional layer applies filters to the input image to extract features, the Pooling layer downsamples the image to reduce computation, and the fully connected layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent.

**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**

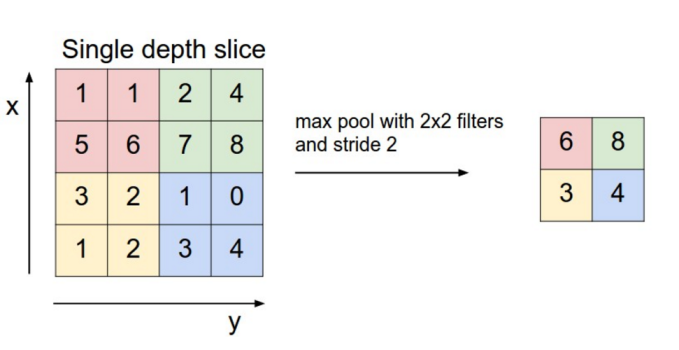
A complete Convolution Neural Networks architecture is also known as covnets. A covnets is a sequence of layers, and every layer transforms one volume to another through a differentiable function.

**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.

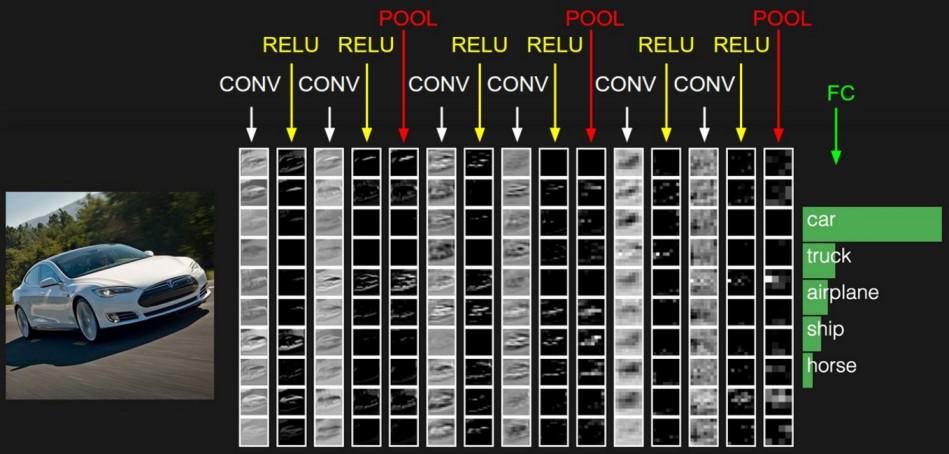
1. **Convolutional Layers:**This is the layer, which is used to extract the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are smaller matrices usually 2×2, 3×3, or 5×5 shape. it slides over the input image data and 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.



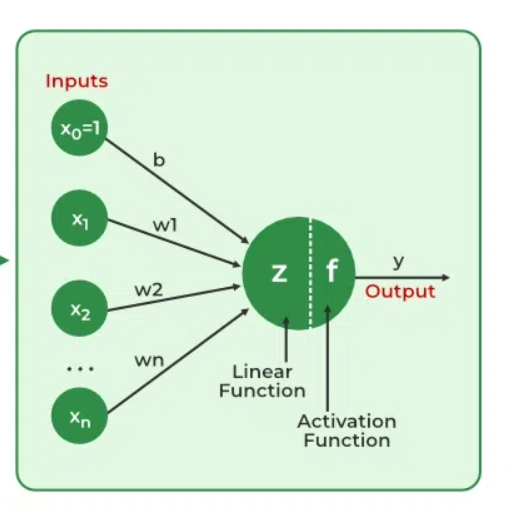
**Output Layer:** The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class

## **Neural Networks Algorithm**

[Neural networks](https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/) extract identifying features from data, lacking pre-programmed understanding. Network components include neurons, connections, weights, biases, propagation functions, and a learning rule. Neurons receive inputs, governed by thresholds and activation functions. Connections involve weights and biases regulating information transfer. Learning, adjusting weights and biases, occurs in three stages: input computation, output generation, and iterative refinement enhancing the network’s proficiency in diverse tasks.

These include:

1. The neural network is simulated by a new environment.
2. Then the free parameters of the neural network are changed as a result of this simulation.
3. The neural network then responds in a new way to the environment because of the changes in its free parameters.



### Importance of Neural Networks

The ability of neural networks to identify patterns, solve intricate puzzles, and adjust to changing surroundings is essential. Their capacity to learn from data has far-reaching effects, ranging from revolutionizing technology like [natural language processing](https://www.geeksforgeeks.org/natural-language-processing-overview/) and self-driving automobiles to automating decision-making processes and increasing efficiency in numerous industries. The development of artificial intelligence is largely dependent on neural networks, which also drive innovation and influence the direction of technology.

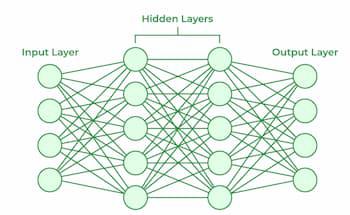
## How does Neural Networks work?

Let’s understand with an example of how a neural network works:

Consider a neural network for email classification. The input layer takes features like email content, sender information, and subject. These inputs, multiplied by adjusted weights, pass through hidden layers. The network, through training, learns to recognize patterns indicating whether an email is spam or not. The output layer, with a binary activation function, predicts whether the email is spam (1) or not (0). As the network iteratively refines its weights through backpropagation, it becomes adept at distinguishing between spam and legitimate emails, showcasing the practicality of neural networks in real-world applications like email filtering.

### Working of a Neural Network

Neural networks are complex systems that mimic some features of the functioning of the human brain. It is composed of an input layer, one or more hidden layers, and an output layer made up of layers of artificial neurons that are coupled. The two stages of the basic process are called backpropagation and [forward propagation](https://www.geeksforgeeks.org/deep-neural-net-with-forward-and-back-propagation-from-scratch-python/).



#### Forward Propagation

* **Input Layer:**Each feature in the input layer is represented by a node on the network, which receives input data.
* **Weights and Connections:** The weight of each neuronal connection indicates how strong the connection is. Throughout training, these weights are changed.
* **Hidden Layers:** Each hidden layer neuron processes inputs by multiplying them by weights, adding them up, and then passing them through an activation function. By doing this, non-linearity is introduced, enabling the network to recognize intricate patterns.
* **Output:** The final result is produced by repeating the process until the output layer is reached.

#### Backpropagation

* **Loss Calculation:** The network’s output is evaluated against the real goal values, and a loss function is used to compute the difference. For a regression problem, the [Mean Squared Error](https://www.geeksforgeeks.org/python-mean-squared-error/) (MSE) is commonly used as the cost function.  
  **Loss Function:**
* **Gradient Descent:**Gradient descent is then used by the network to reduce the loss. To lower the inaccuracy, weights are changed based on the derivative of the loss with respect to each weight.
* **Adjusting weights:** The weights are adjusted at each connection by applying this iterative process, or [backpropagation](https://www.geeksforgeeks.org/backpropagation-in-data-mining/), backward across the network.
* **Training:**During training with different data samples, the entire process of forward propagation, loss calculation, and backpropagation is done iteratively, enabling the network to adapt and learn patterns from the data.
* **Actvation Functions:** Model non-linearity is introduced by activation functions like the [rectified linear unit](https://www.geeksforgeeks.org/activation-functions-neural-networks/) (ReLU) or [sigmoid](https://www.geeksforgeeks.org/derivative-of-the-sigmoid-function/). Their decision on whether to “fire” a neuron is based on the whole weighted input.

## **Learning of a Neural Network**

### **1. Learning with supervised learning**

In [supervised learning](https://www.geeksforgeeks.org/supervised-machine-learning/), the neural network is guided by a teacher who has access to both input-output pairs. The network creates outputs based on inputs without taking into account the surroundings. By comparing these outputs to the teacher-known desired outputs, an error signal is generated. In order to reduce errors, the network’s parameters are changed iteratively and stop when performance is at an acceptable level.

### **2. Learning with Unsupervised learning**

Equivalent output variables are absent in [unsupervised learning](https://www.geeksforgeeks.org/supervised-unsupervised-learning/). Its main goal is to comprehend incoming data’s (X) underlying structure. No instructor is present to offer advice. Modeling data patterns and relationships is the intended outcome instead. Words like regression and classification are related to supervised learning, whereas unsupervised learning is associated with clustering and association.

### **3. Learning with Reinforcement Learning**

Through interaction with the environment and feedback in the form of rewards or penalties, the network gains knowledge. Finding a policy or strategy that optimizes cumulative rewards over time is the goal for the network. This kind is frequently utilized in gaming and decision-making applications.

## Types of Neural Networks

There are seven types of neural networks that can be used.

* **Feedforward Neteworks:** A [feedforward neural network](https://www.geeksforgeeks.org/difference-between-feed-forward-neural-networks-and-recurrent-neural-networks/) is a simple artificial neural network architecture in which data moves from input to output in a single direction. It has input, hidden, and output layers; feedback loops are absent. Its straightforward architecture makes it appropriate for a number of applications, such as regression and pattern recognition.
* **Multilayer Perceptron (MLP):** [MLP](https://www.geeksforgeeks.org/difference-between-multilayer-perceptron-and-linear-regression/) is a type of feedforward neural network with three or more layers, including an input layer, one or more hidden layers, and an output layer. It uses nonlinear activation functions.
* **Convolutional Neural Network (CNN):** A [Convolutional Neural Network](https://www.geeksforgeeks.org/introduction-convolution-neural-network/)(CNN) is a specialized artificial neural network designed for image processing. It employs convolutional layers to automatically learn hierarchical features from input images, enabling effective image recognition and classification. CNNs have revolutionized computer vision and are pivotal in tasks like object detection and image analysis.
* **Recurrent Neural Network (RNN):**An artificial neural network type intended for sequential data processing is called a [Recurrent Neural Network](https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/)(RNN). It is appropriate for applications where contextual dependencies are critical, such as time series prediction and natural language processing, since it makes use of feedback loops, which enable information to survive within the network.
* **Long Short-Term Memory (LSTM):**[LSTM](https://www.geeksforgeeks.org/long-short-term-memory-networks-explanation/) is a type of RNN that is designed to overcome the vanishing gradient problem in training RNNs. It uses memory cells and gates to selectively read, write, and erase information.

**5.2 Sample Code:**

import numpy as np

import pandas as pd

import shap *#loading SHAP tool for XAI based explanation*

from sklearn.neural\_network import MLPClassifier

from sklearn.preprocessing import LabelEncoder

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

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score

from sklearn.metrics import f1\_score

from sklearn.metrics import precision\_score

from sklearn.metrics import recall\_score

from xgboost import XGBClassifier

from keras\_dgl.layers import GraphCNN *#loading GNN class*

import keras.backend as K

from keras.regularizers import l2

from keras.utils.np\_utils import to\_categorical

from keras.layers import MaxPooling2D

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.layers import Convolution2D

from keras.models import Sequential, load\_model

import os

from keras.callbacks import ModelCheckpoint

import pickle

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import roc\_curve

from sklearn.metrics import roc\_auc\_score

from sklearn import metrics

import seaborn as sns

*#load and display dataset values*

dataset = pd.read\_csv('Dataset/drug200.csv')

dataset

*#visualize graph of different drug found in dataset*

labels = np.unique(dataset['Drug'])

label = dataset.groupby('Drug').size()

label.plot(kind="bar", figsize=(4,3))

plt.xlabel('Drug Type')

plt.ylabel('Number of Records')

plt.xticks(rotation=90)

plt.title("Drug Graph")

plt.show()

*#dataset processing like converting non-numeric data to numeric data*

label\_encoder = []

columns = dataset.columns

types = dataset.dtypes.values

for i in range(len(types)):

name = types[i]

if name == 'object': *#finding column with object type*

le = LabelEncoder()

dataset[columns[i]] = pd.Series(le.fit\_transform(dataset[columns[i]].astype(str)))*#encode all str columns to numeric*

label\_encoder.append([columns[i], le])

dataset.fillna(0, inplace = True)

dataset

*#dataset shuffling and splitting data into train and test*

Y = dataset['Drug'].ravel()

dataset.drop(['Drug'], axis = 1,inplace=True)

X = dataset.values

indices = np.arange(X.shape[0])

np.random.shuffle(indices)*#shuffling dataset values*

X = X[indices]

Y = Y[indices]

scaler = StandardScaler()

X = scaler.fit\_transform(X)*#features normalization*

*#split dataset into train and test*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size = 0.4)

print("Total records found in dataset = "+str(X.shape[0]))

print("Total features found in dataset= "+str(X.shape[1]))

print("80% dataset for training : "+str(X\_train.shape[0]))

print("20% dataset for testing : "+str(X\_test.shape[0]))

*#define global variables to save accuracy and other metrics*

accuracy = []

precision = []

recall = []

fscore = []

*#function to calculate all metrics*

def calculateMetrics(algorithm, testY, predict):

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

accuracy.append(a)

precision.append(p)

recall.append(r)

fscore.append(f)

print(algorithm+" Accuracy : "+str(a))

print(algorithm+" Precision : "+str(p))

print(algorithm+" Recall : "+str(r))

print(algorithm+" FSCORE : "+str(f))

cm = metrics.confusion\_matrix(testY,predict)

plt.figure(figsize =(6, 3))

ax = sns.heatmap(cm,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()

*#training XGBOOST algorithm on 80% training data and then predicting on 20% test data*

xgc = XGBClassifier(n\_estimators=2)

xgc.fit(X\_train, y\_train)

*#performing prediction on test data*

predict = xgc.predict(X\_test)

*#calling function to calculate accuracy on predicted data*

calculateMetrics("XGBoost", y\_test, predict)

*#training graphCNN algorithm*

y\_train1 = to\_categorical(y\_train)

y\_test1 = to\_categorical(y\_test)

*#Create GNN model to detect fault from all services*

graph\_conv\_filters = np.eye(1)

graph\_conv\_filters = K.constant(graph\_conv\_filters)

graph\_model = Sequential()

graph\_model.add(GraphCNN(128, 1, graph\_conv\_filters, input\_shape=(X\_train.shape[1],), activation='elu', kernel\_regularizer=l2(5e-4)))

graph\_model.add(GraphCNN(64, 1, graph\_conv\_filters, input\_shape=(X\_train.shape[1],), activation='elu', kernel\_regularizer=l2(5e-4)))

graph\_model.add(GraphCNN(1, 1, graph\_conv\_filters, input\_shape=(X\_train.shape[1],), activation='elu', kernel\_regularizer=l2(5e-4)))

graph\_model.add(Dense(units = 256, activation = 'elu'))

graph\_model.add(Dense(units = y\_train1.shape[1], activation = 'softmax'))

graph\_model.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])

if os.path.exists("model/gcnn\_weights.h5") == False:

hist = graph\_model.fit(X\_train, y\_train1, batch\_size=1, epochs=50, validation\_data = (X\_test, y\_test1), verbose=1)

graph\_model.save\_weights("model/gcnn\_weights.h5")

else:

graph\_model.load\_weights("model/gcnn\_weights.h5")

*#perform prediction on test data of all services and calculate accuracy and other metrics*

pred = []

for i in range(len(X\_test)):

temp = []

temp.append(X\_test[i])

temp = np.asarray(temp)

predict = graph\_model.predict(temp, batch\_size=1)

predict = np.argmax(predict)

pred.append(predict)

y\_tested = np.argmax(y\_test1, axis=1)

predict = np.asarray(pred)

*#calling function to calculate accuracy on predicted data*

calculateMetrics("Graph Model", y\_tested, predict)

*#training neural network algorithm*

nn = MLPClassifier(max\_iter=800)

nn.fit(X\_train, y\_train)

predict = nn.predict(X\_test)

*#calling function to calculate accuracy on predicted data*

calculateMetrics("Neural Network", y\_test, predict)

*#training extension Convolution Neural Network 2D algortihm as extension*

X\_train1 = np.reshape(X\_train, (X\_train.shape[0], X\_train.shape[1], 1, 1))

X\_test1 = np.reshape(X\_test, (X\_test.shape[0], X\_test.shape[1], 1, 1))

cnn\_model = Sequential()

*#creating cnn2d layer with 32 neurons of 1 X 1 matrix to filter dataset 32 times*

cnn\_model.add(Convolution2D(32, (1, 1), input\_shape = (X\_train1.shape[1], X\_train1.shape[2], X\_train1.shape[3]), activation = 'relu'))

*#max layer tio collect relevant filtered features from CNN layer*

cnn\_model.add(MaxPooling2D(pool\_size = (1, 1)))

cnn\_model.add(Convolution2D(32, (1, 1), activation = 'relu'))

cnn\_model.add(MaxPooling2D(pool\_size = (1, 1)))

*#defining output layer*

cnn\_model.add(Flatten())

cnn\_model.add(Dense(units = 256, activation = 'relu'))

cnn\_model.add(Dense(units = y\_train1.shape[1], activation = 'softmax'))

*#compiling the model*

cnn\_model.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])

*#training and loading model*

if os.path.exists("model/cnn\_weights.hdf5") == False:

model\_check\_point = ModelCheckpoint(filepath='model/cnn\_weights.hdf5', verbose = 1, save\_best\_only = True)

hist = cnn\_model.fit(X\_train1, y\_train1, batch\_size = 8, epochs = 50, validation\_data=(X\_test1, y\_test1), callbacks=[model\_check\_point], verbose=1)

f = open('model/cnn\_history.pckl', 'wb')

pickle.dump(hist.history, f)

f.close()

else:

cnn\_model.load\_weights("model/cnn\_weights.hdf5")

*#performing prediction on test data*

predict = cnn\_model.predict(X\_test1)

predict = np.argmax(predict, axis=1)

y\_test1 = np.argmax(y\_test1, axis=1)

*#calling function to calculate accuracy on predicted data*

calculateMetrics("Extension CNN2D", y\_test1, predict)

*#comparison graph between all algorithms*

df = pd.DataFrame([['XGBoost','Accuracy',accuracy[0]],['XGBoost','Precision',precision[0]],['XGBoost','Recall',recall[0]],['XGBoost','FSCORE',fscore[0]],

['Graph Model','Accuracy',accuracy[1]],['Graph Model','Precision',precision[1]],['Graph Model','Recall',recall[1]],['Graph Model','FSCORE',fscore[1]],

['Neural Network','Accuracy',accuracy[2]],['Neural Network','Precision',precision[2]],['Neural Network','Recall',recall[2]],['Neural Network','FSCORE',fscore[2]],

['Extension CNN2D','Accuracy',accuracy[3]],['Extension CNN2D','Precision',precision[3]],['Extension CNN2D','Recall',recall[3]],['Extension CNN2D','FSCORE',fscore[3]],

],columns=['Parameters','Algorithms','Value'])

df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar', figsize=(6, 3))

plt.title("All Algorithms Performance Graph")

plt.show()

*#now explain model prediction using features impact and shap tool*

*#here we are uisng force plot explanation to explain about features which is contributing most for the model to make*

*#correct prediction*

shap.initjs()

explainer = shap.TreeExplainer(xgc, X\_train)

*# Explain the predictions of your model*

shap\_values = explainer.shap\_values(X\_train)

*# Plot the SHAP values*

shap.plots.force(explainer.expected\_value[0], shap\_values[0], feature\_names=['Age','Sex','BP','Cholesterol','Na\_to\_K'])

*#summarry plot to explain names of features which is contributing most for the algorithm to make correct prediction*

shap.summary\_plot(shap\_values[0], X\_train, feature\_names=['Age','Sex','BP','Cholesterol','Na\_to\_K'])

*#violin plot to explain names of features which is contributing most for the algorithm to make correct prediction*

shap.plots.violin(shap\_values[0], feature\_names=['Age','Sex','BP','Cholesterol','Na\_to\_K'])

*#decision plot to explain names of features which is contributing most for the algorithm to make correct prediction*

shap.decision\_plot(explainer.expected\_value[0], shap\_values[0], feature\_names=['Age','Sex','BP','Cholesterol','Na\_to\_K'])

*#predict drug based on patient condition*

testData = pd.read\_csv("Dataset/testData.csv")*#load test data*

temps = testData.values

for i in range(len(label\_encoder)-1):

temp = label\_encoder[i]

name = temp[0]

le = temp[1]

testData[name] = pd.Series(le.transform(testData[name].astype(str)))*#encode all str columns to numeric*

testData.fillna(0, inplace = True)*#replace missing values*

testData = testData.values

X = scaler.transform(testData)*#normalize test data*

X = np.reshape(X, (X.shape[0], X.shape[1], 1, 1))*#reshape test data as cnn2d format*

predict = cnn\_model.predict(X)*#predict on test data and then display predicted drug*

for i in range(len(predict)):

print("Test Data = "+str(temps[i])+" Predicted Drug = "+labels[np.argmax(predict[i])])

print()

**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 modifying 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.

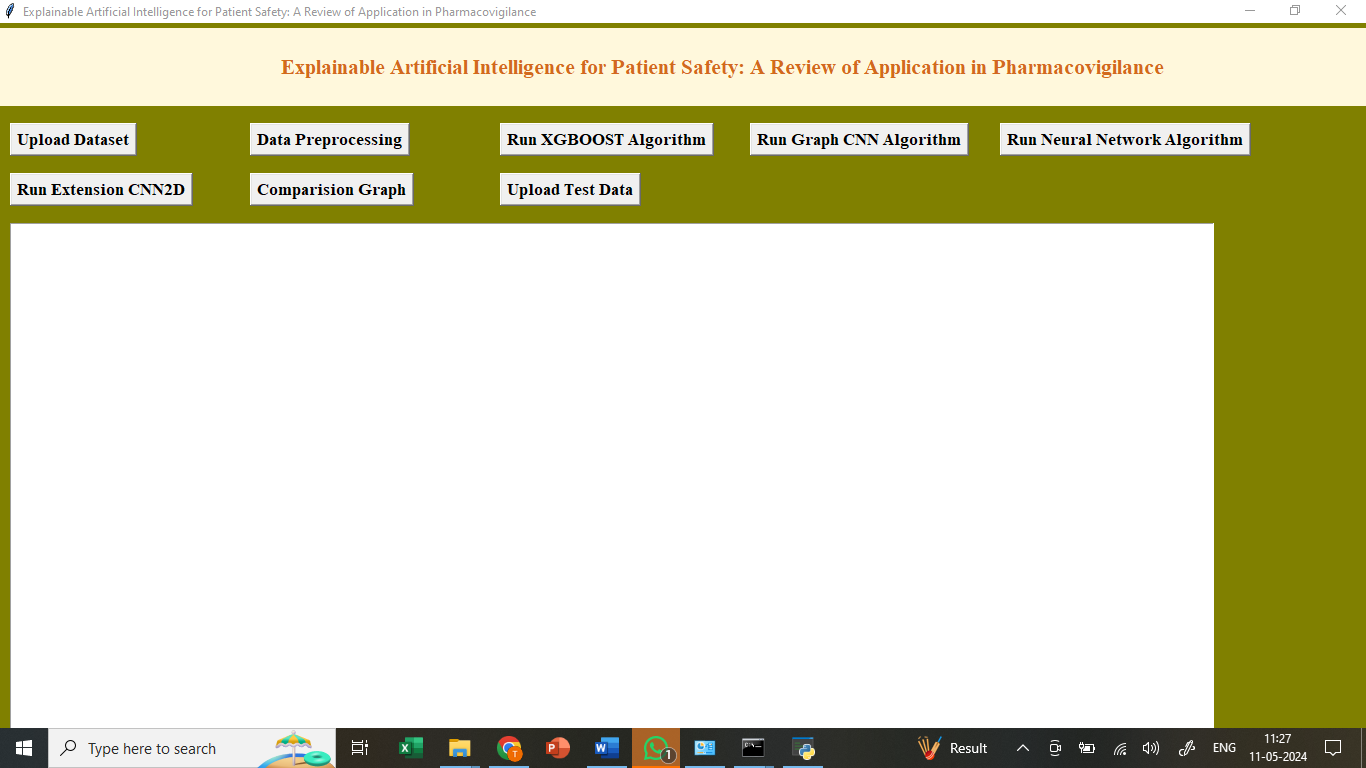
**Acceptance Testing**

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.

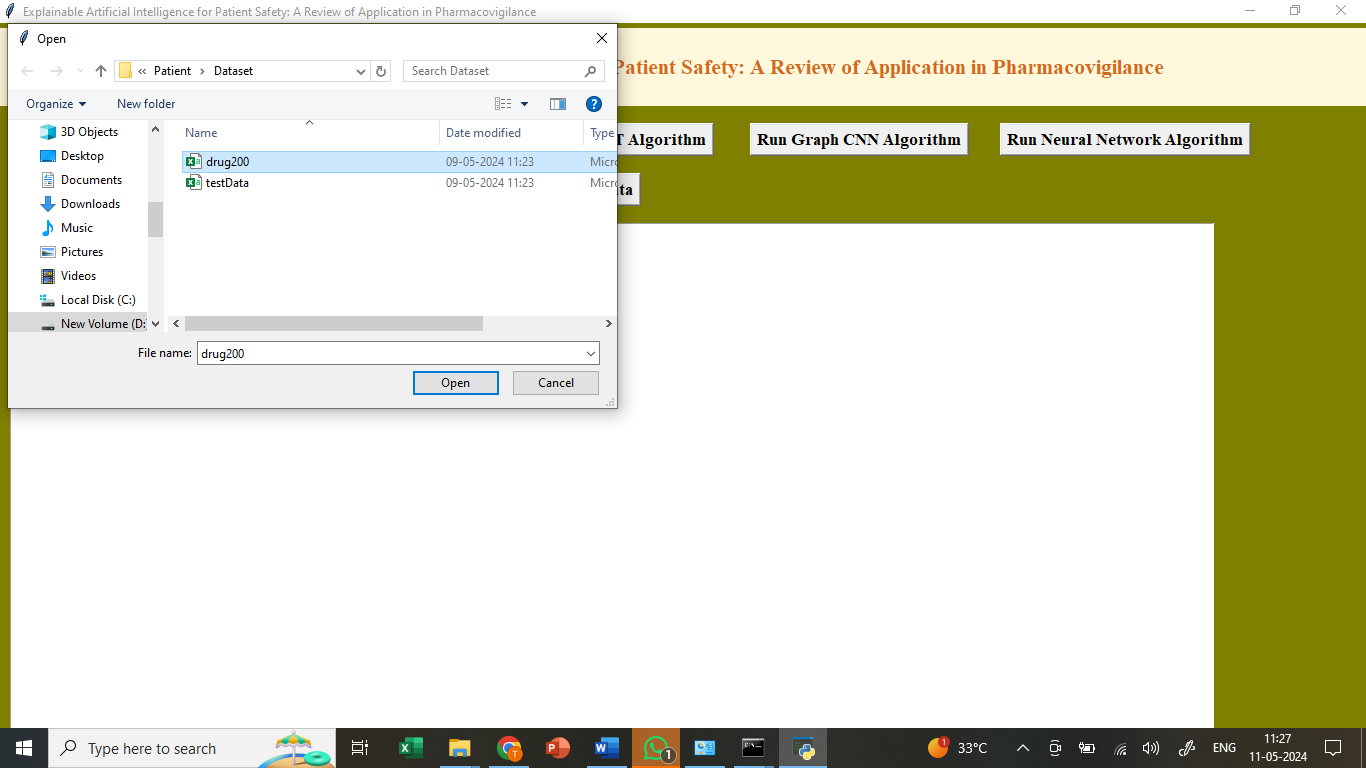
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | | **Actual** |
| 01 | Upload dataset | Verify dataset loaded or not | If dataset may not uploaded | we cannot do any further operations | we can do further operations | | High | High |
| 02 | Preprocess Dataset | Verify Preprocess Dataset or not | If Dataset may not Preprocess | we cannot do any further operations | we can do further operations | | High | High |
| 03 | Run Xgboost Algorithm | Verify Xgboost Algorithm runned or not | If Xgboost Algorithm not run | we cannot do any further operations | we can do further operations | | High | High |
| 04 | Run Graph CNN Algorithm | Verify Graph Cnn Algorithm runned or not | If Graph CNN Algorithm not Run | We cannot run  operations | we can do further operations | | High | High |
| 05 | Run Neural Network algorithm | Verify Neural Network algorithm runned or not | If Neural Network algorithm not run | We cannot run further  operations | we can do further operations | | High | High |
| 06 | Run Extension CNN2D Algortihm | Verify Extension CNN2D Algorithm runned or not | If Extension CNN2D Algorithm nor run | We cannot run further  operations | we can do further operations | | high | high |
| 07 | Comparison Graph | Verify Comparison graph obtained or not | If comparison graph is not obtained | We cannot run further  operations | we can do further operations | | high | high |
| 08 | Upload test data | Verify test data is uploaded or not | If test data is not uploaded | We cannot run further  operations | we can do further operations | | high | high |

**7.SCREENSHOTS:**

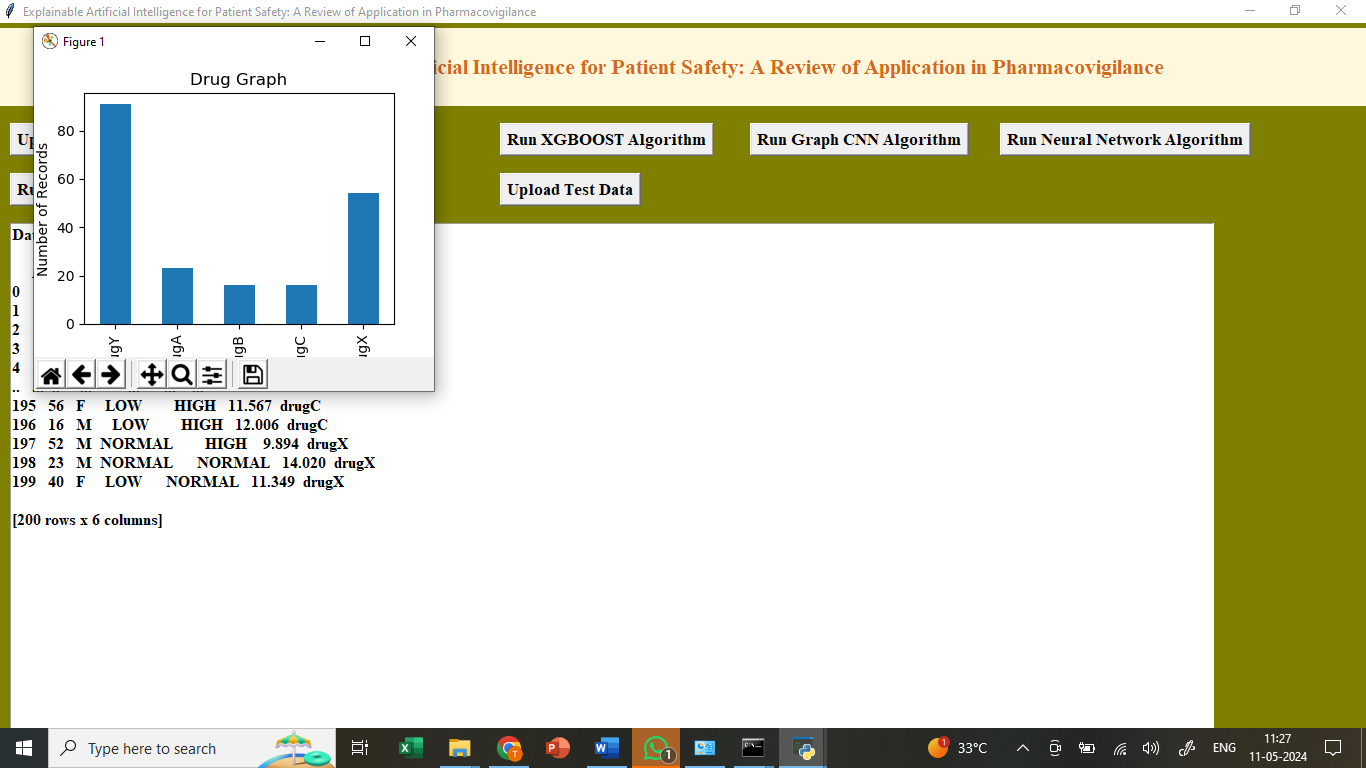
To Run the application Click on “run.bat” file from the file location.



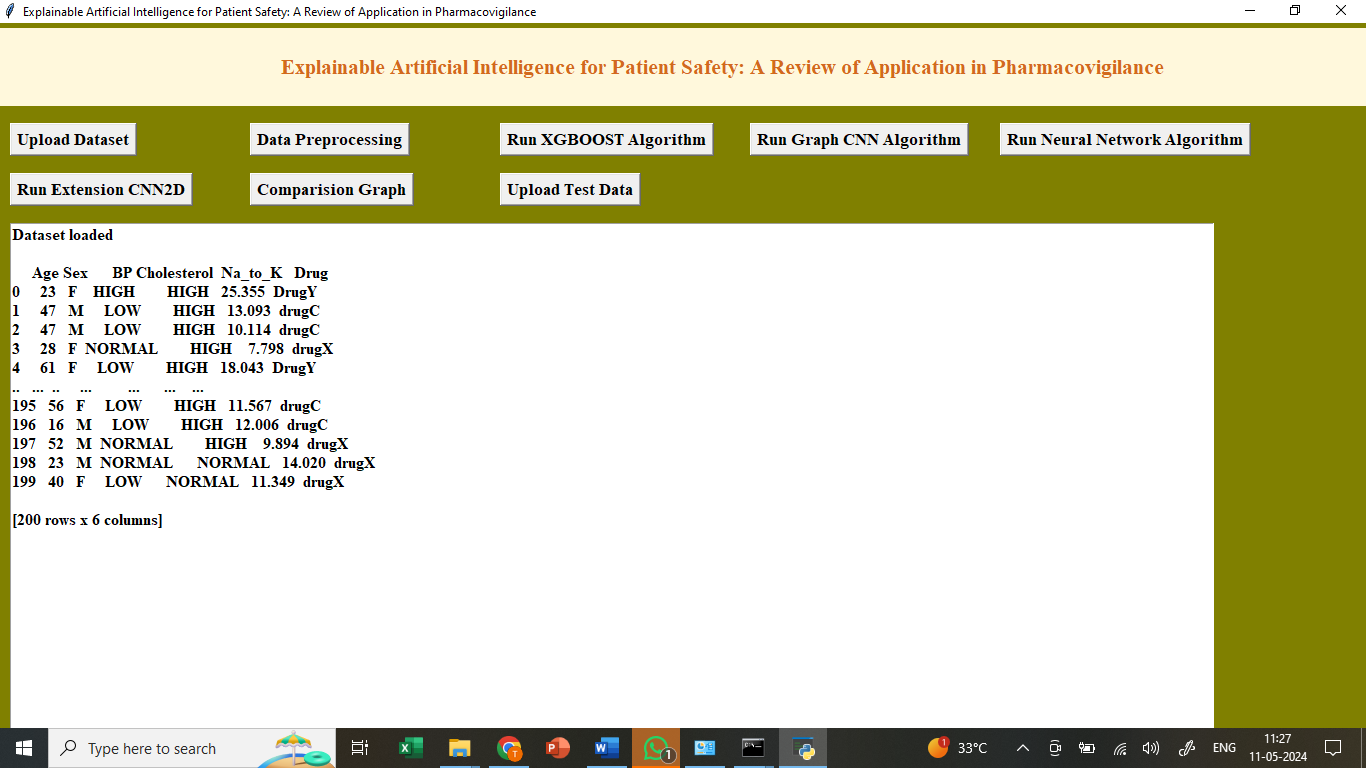
In the above screen we got Tkinter Output Window. Now Click On the “Upload Dataset” button to upload the dataset to the application.



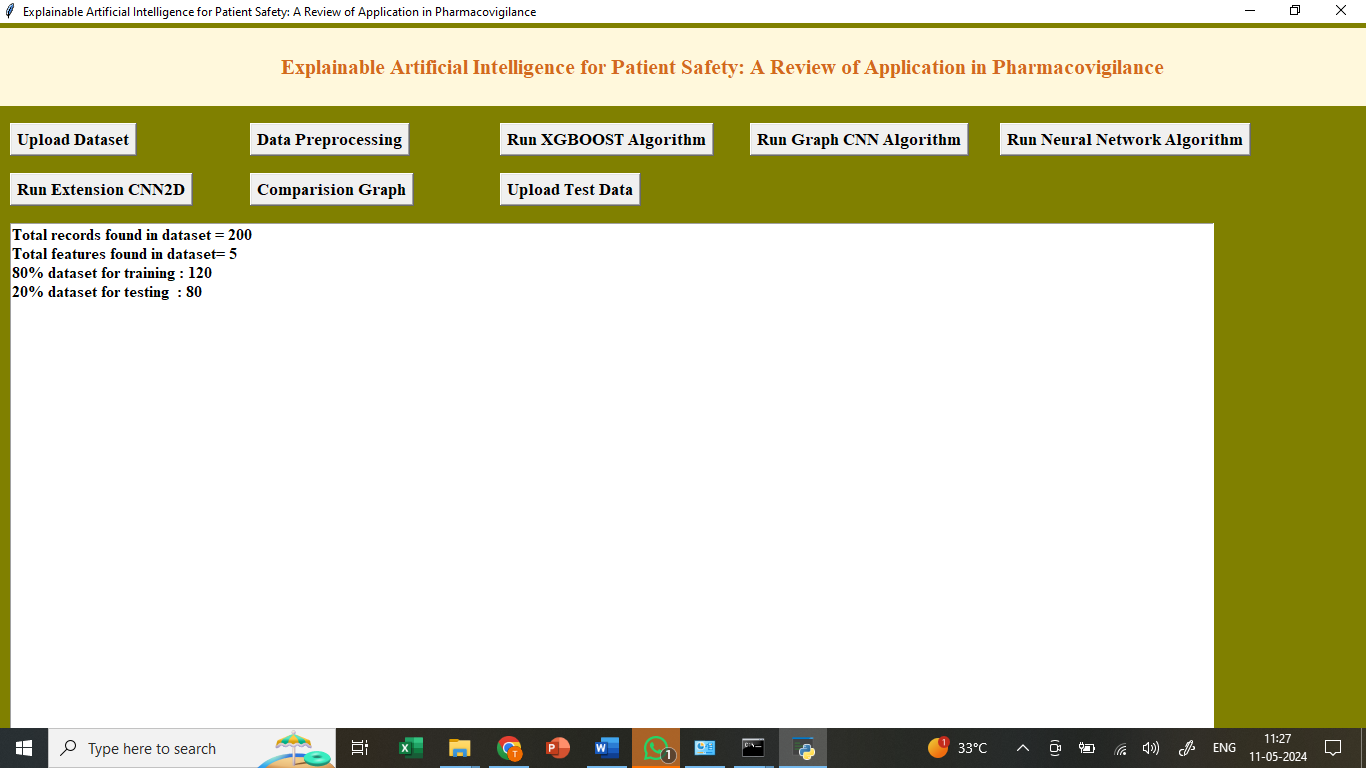
In the above screen selecting the dataset and Uploading to the application for further operations.



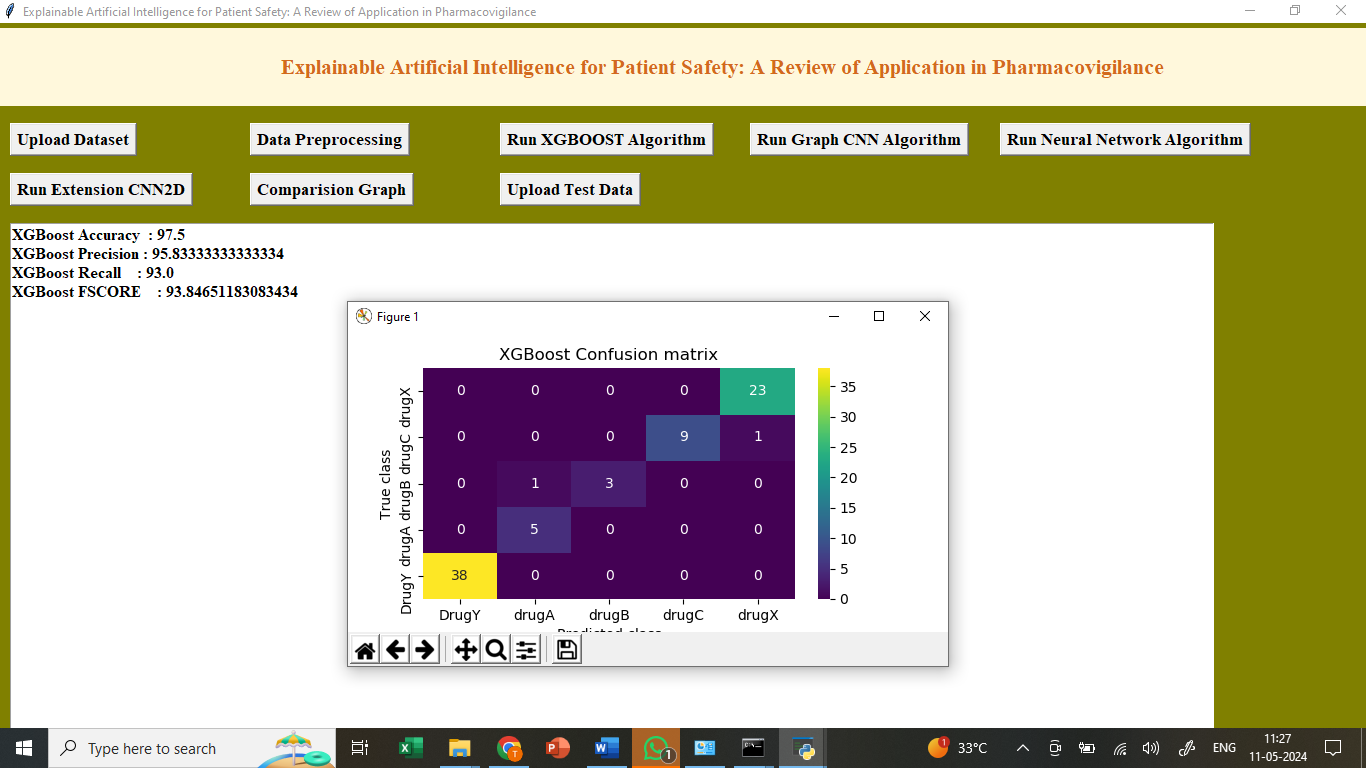
In the above screen we can see the dataset uploaded and we got dataset graph generated displaying types of Drugs and Number of records found in dataset as graph format.



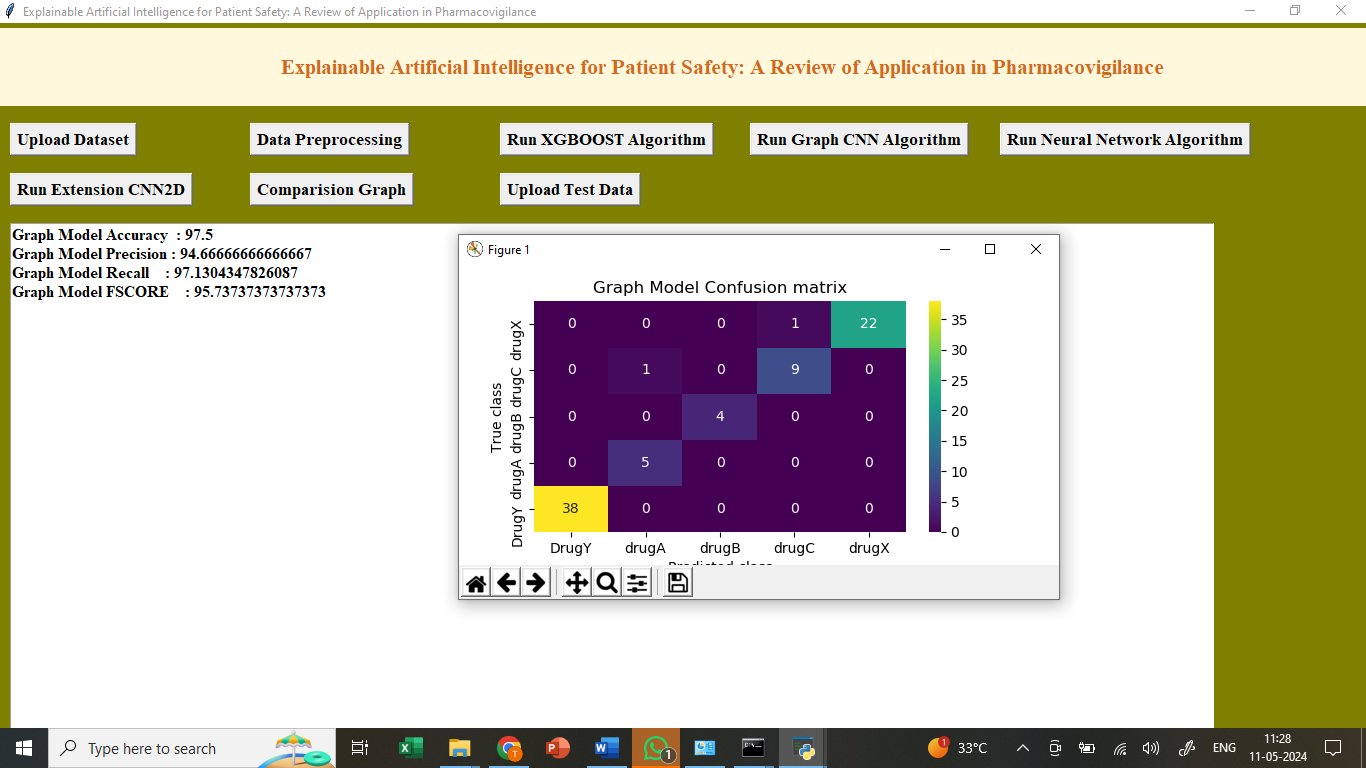
In the above screen we can see Datasets are loaded. And now click in “Dataset Preprocessing” because Dataset contains both numeric and non-numeric values so in below screen employing label encoder class to convert non-numeric data to numeric data and after conversion can see all values are in numeric format.



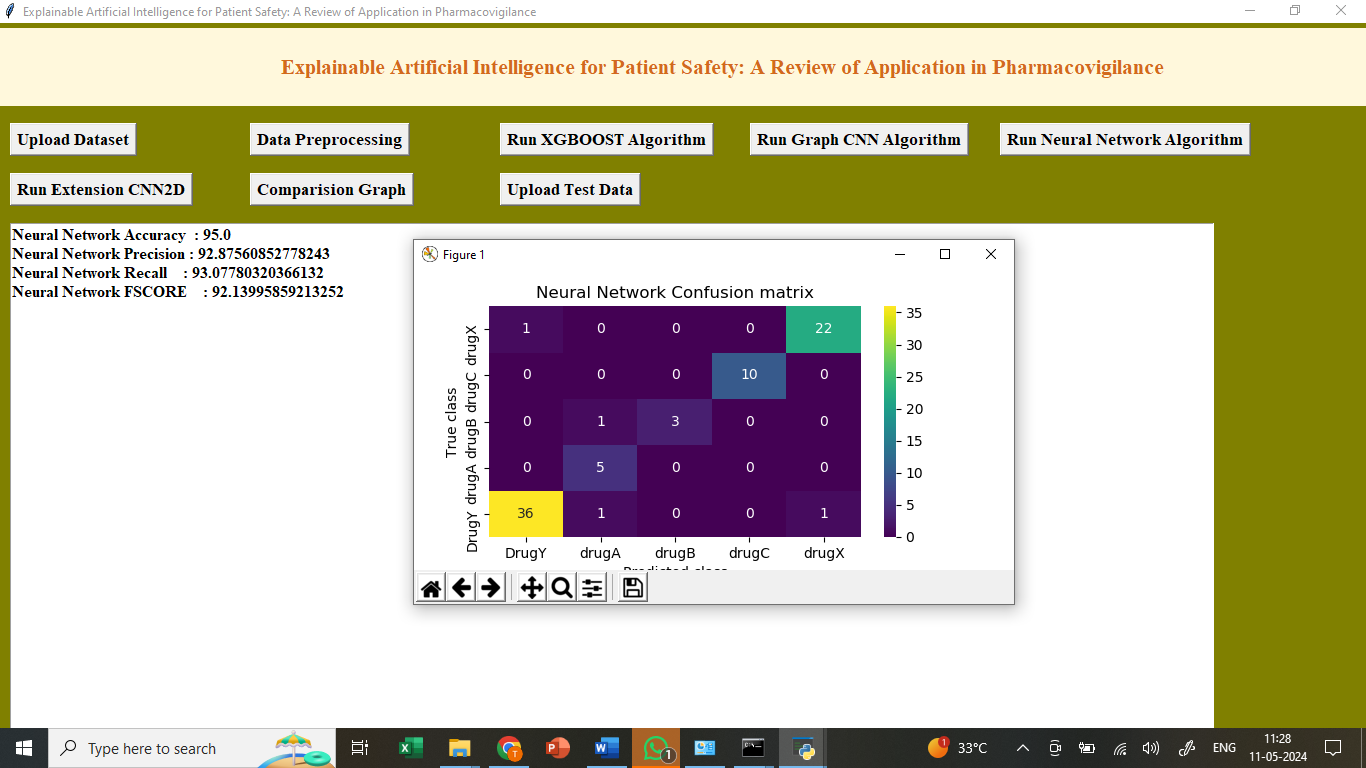
In above screen applying processing like data normalization, shuffling and splitting into train and test part with a ratio of 80:20. 80% for training and 20% for testing. Here Data-preprocessing Completed now click on “Run Xgboost Algorithm”.



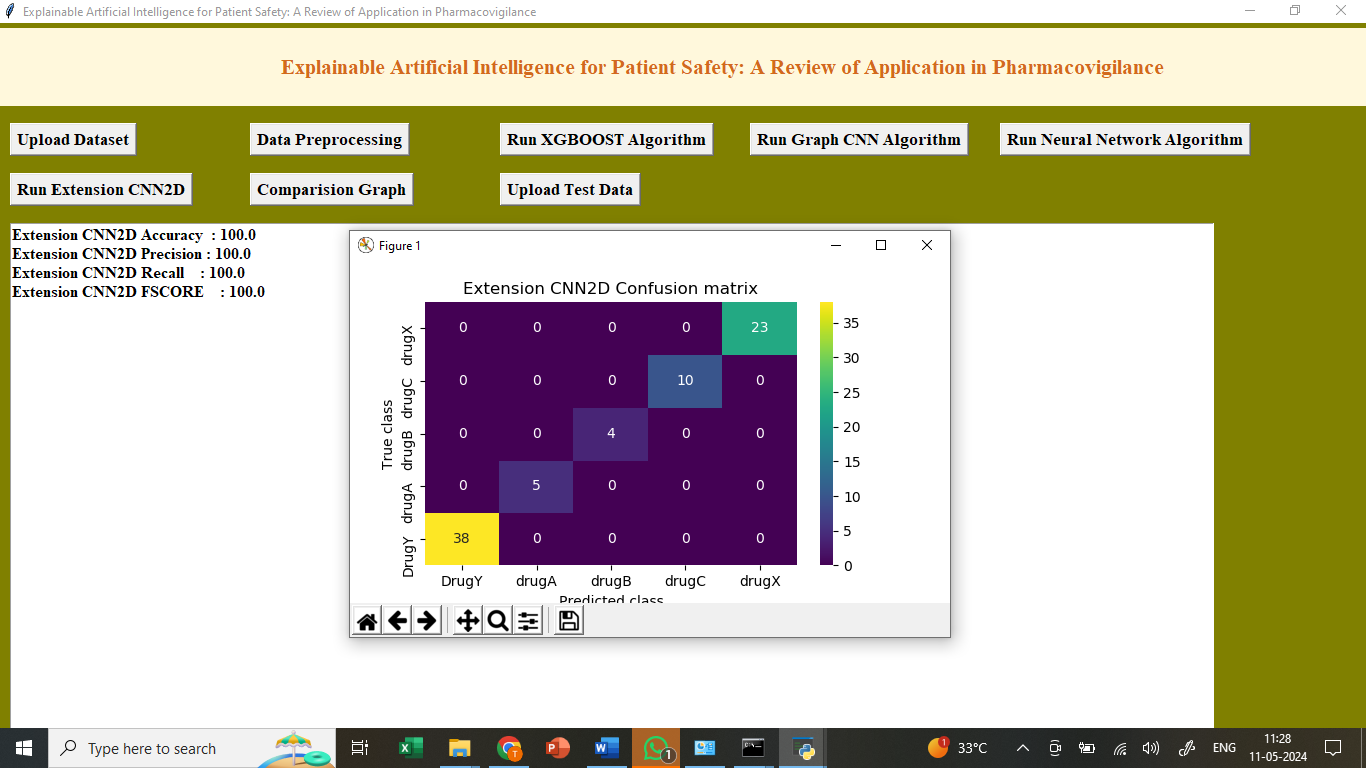
In above screen training XGBOOST algorithm as tree model and it got 95% accuracy and can see other metrics like precision, recall and FSCORE. In above XGBOOST confusion matrix graph x-axis represents ‘Predicted Labels’ and y-axis represents True labels and then all different colour boxes in diagnol represents correct prediction count and all blue boxes represents incorrect prediction count which are very few. Now click on “Run Graph CNN Algorithm ”.



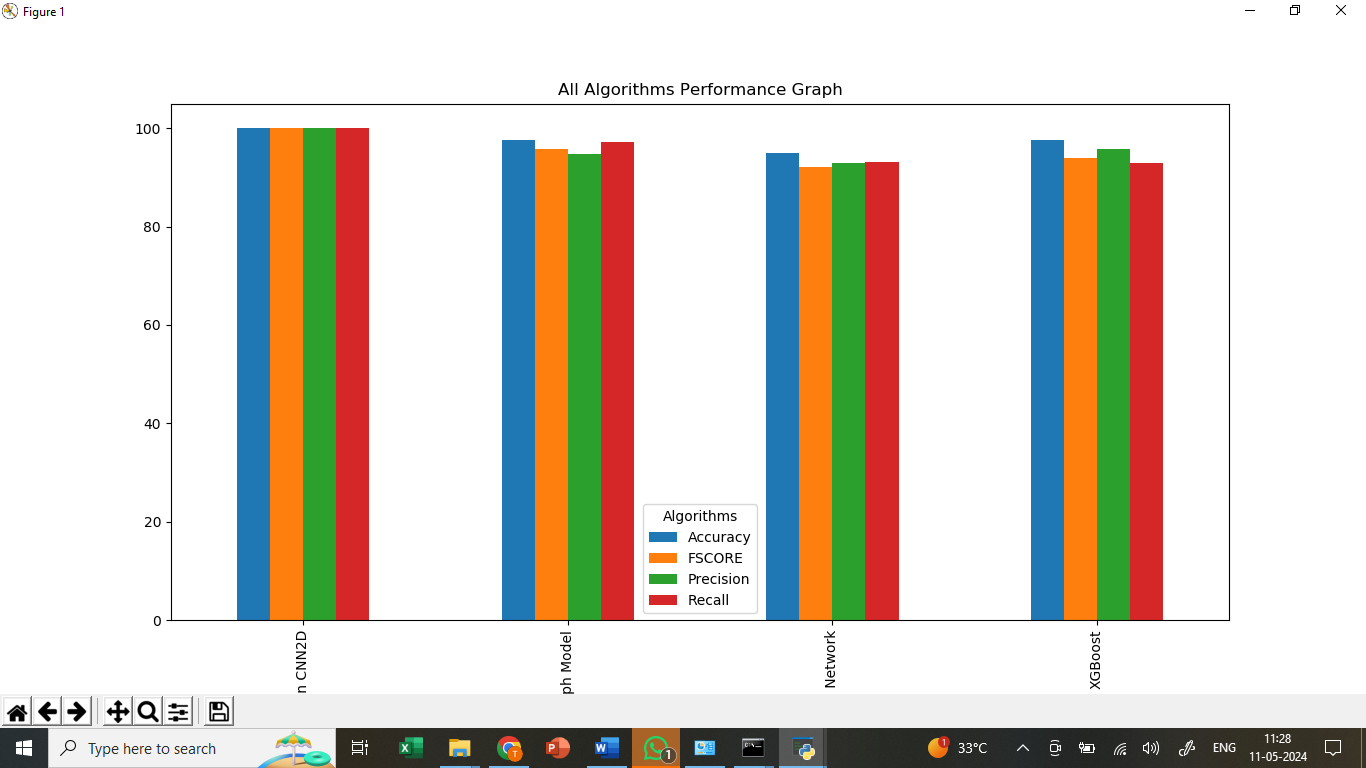
In above screen training GRAPHCNN algorithm and after execution will get above output. GRAPHCNN got 93% accuracy and can see all other metrics output. Now click on “Run Neural Network Algorithm ”.



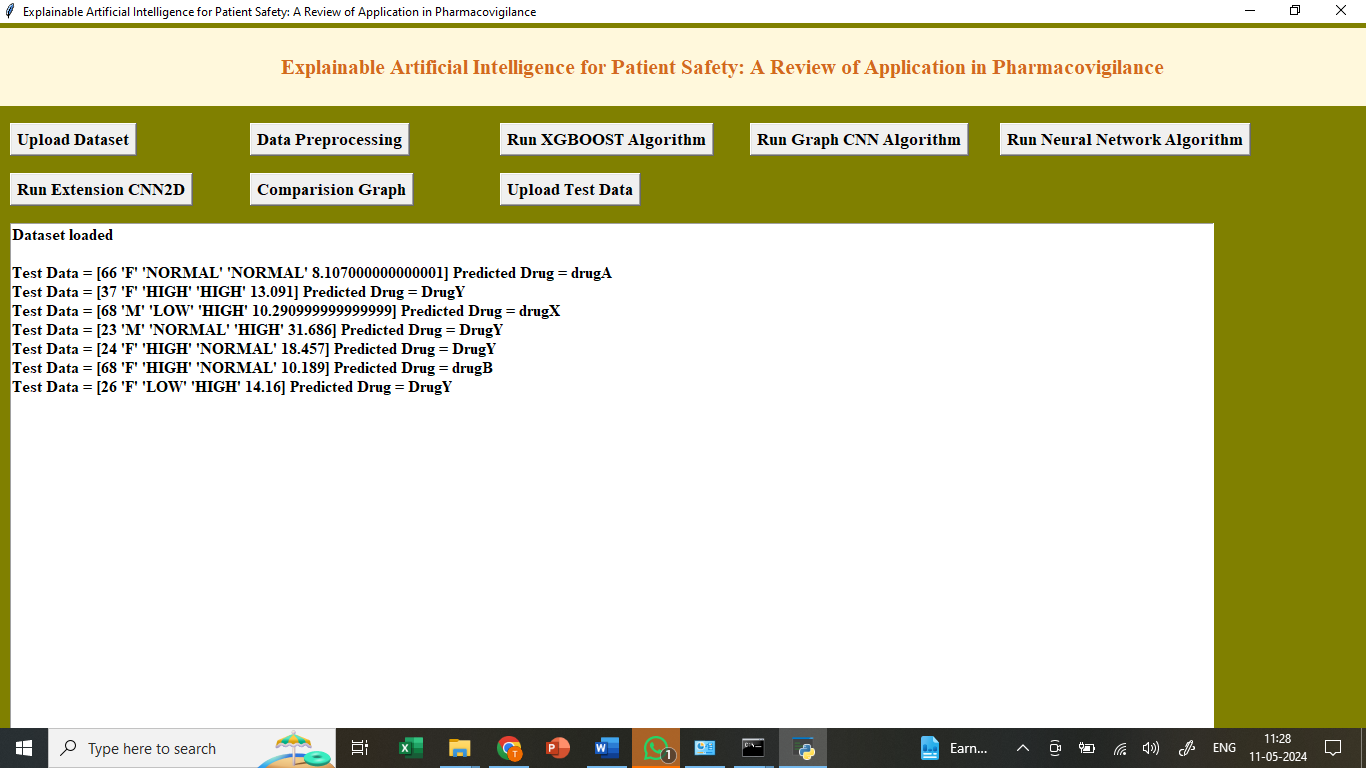
In above screen neural network got 91% accuracy and can see all other metrics output. Now click on “Run Extension CNN2D Algorithm ”.



In above screen training extension CNN2d algorithm and extension CNN2D got 98% accuracy and can see all other metrics in output. Now Click on “Comparison Graph” to get below output .



In above screen displaying comparison between all algorithms where x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars and in all algorithms extension CNN2D got high performance. Now Click On “Upload Test Dataset”.



In the above screen we can see the test data uploaded and predicted the output result from the test data.

So by using above implementation we have trained models and then explain about all features which is contributing or impacting model most for correct prediction.

**8.CONCLUSION:**

In this study, we reviewed PV XAI papers and discussed recent research trends and the need for XAI research. Unlike other areas where XAI and AI are developing together, PV XAI research is still in its infancy. There are not many papers on PV XAI and the methodology is limited to a few models. However, studies are slowly beginning to show the potential of XAI research for medication monitoring and patient safety, collecting ADR and ADE information, extracting drug-drug interactions, and predicting drug treatment effects. As in other areas, as awareness of XAI methods grows, we expect to see AI used in pharmacovigilance and patient safety in many more ways in the coming years than those identified in this review, and the positive potential of XAI for drug therapy, ADRs and interactions is very promising. However, it is clear that the growth of this field may be limited by the lack of validated and established uses of XAI in real-world healthcare settings, and this is an area that requires further investigation. Therefore, the challenges and future prospects of XAIs in pharmacovigilance should be discussed with continued interest.

**9. REFERENCES:**

Kumar A, Kaur A, Kumar M. Face detection techniques: a review[J].

Artificial Intelligence Review, 2019, 52(2): 927-948.

2. Sousa M J, Moutinho A, Almeida M. Wildfire detection using transfer

learning on augmented datasets[J]. Expert Systems with Applications,

2020, 142: 112975.

3. Voulodimos A, Doulamis N, Doulamis A, et al. Deep learning for

computer vision: A brief review[J]. Computational intelligence and

neuroscience, 2018, 2018.

4. Tang B, Kong J Y, Wu S Q . Review of surface defect detection based

on machine vision[J]. Journal of Image and Graphics, 2017, 22(12): 1640-

1663.

5. Yin S, Ren Y J, Liu T, et al. Review on application of machine vision in

modern automobile manufacturing[J]. Acta Optica Sinica, 2018, 38(8):

0815001.

6. Min F, Lu T. Exploration and practice of machine vision course

teaching for production practice[J]. Computer Education, 2017(10):41-43.

7. Wang Z, Xiao G, Liu H. Teaching Reform of the Course of Machine

Vision Principles and Applications for Professional Degree

Postgraduates[J]. Education Teaching Forum, 2021, 3(11): 37-40.

8. Shao T, Tang J, Liao Y. Comprehensive experiment system of cocoon

sorting based on machine vision[J]. Experimental Technology and

Management, 2020, v.37;No.289(09):125-127+132.

9. Han Z, Liu X. Design of machine vision experiment teaching platform

based on Python[J]. Computer Measurement and Control, 2020,

v.28;No.258(03):249-253+258.

10. Sigut J, Castro M, Arnay R, et al. OpenCV basics: a mobile application

to support the teaching of computer vision concepts[J]. IEEE Transactions

on Education, 2020, 63(4): 328-335.

11. Cote M, Albu A B. Teaching Computer Vision and Its Societal Effects:

A Look at Privacy and Security Issues From the Students’

Perspective[C]//2017 IEEE Conference on Computer Vision and Pattern

Recognition Workshops (CVPRW). IEEE, 2017: 1378-1386.

12. Spurlock S, Duvall S. Making computer vision accessible for

undergraduates[J]. Journal of Computing Sciences in Colleges, 2017,

33(2): 215-221.

13. Jiang Y, Li B. Exploration on the teaching reform measure for machine

learning course system of artificial intelligence specialty[J]. Scientific

Programming, 2021, 2021.

14. Wang Y, Stanković M, Smith A, et al. Leader-Follower System in

Convoys:: An Experimental Design Focusing on Computer

Vision[C]//2021 IEEE Sensors Applications Symposium (SAS). IEEE,

2021: 1-6.

15. Shamsoshoara A, Afghah F, Razi A, et al. Aerial Imagery Pile burn

detection using Deep Learning: the FLAME dataset[J]. Computer

Networks, 2021, 193: 108001.

16. Bouguettaya A, Zarzour H, Taberkit A M, et al. A review on early

wildfire detection from unmanned aerial vehicles using deep learning-

based computer vision algorithms[J]. Signal Processing, 2022, 190:

108309.

17. Saldamli G, Deshpande S, Jawalekar K, et al. Wildfire detection using

wireless mesh network[C]//2019 Fourth International Conference on Fog

and Mobile Edge Computing (FMEC). IEEE, 2019: 229-234.

18. Toan N T, Cong P T, Hung N Q V, et al. A deep learning approach for

early wildfire detection from hyperspectral satellite images[C]//2019 7th

International Conference on Robot Intelligence Technology and

Applications (RiTA). IEEE, 2019: 38-45.

19. Jia L, Jiong Y. Multi-information Fusion Flame Detection Based on

Ohta Color Space[J]. Journal of East China University of Science and

Technology (Natural Science Edition) , 2019, 45(6): 962-969.

20. Tan Y, Xie L, Feng H, et al. Flame detection algorithm based on image

processing technology[J]. Laser & Optoelectronics Progress, 2019, 56(16):

161012.

21. Hossain F M A, Zhang Y, Yuan C, et al. Wildfire flame and smoke

detection using static image features and artificial neural network[C]//2019

1st international conference on industrial artificial intelligence (IAI). IEEE,

2019: 1-6.

22. Schultze T, Kempka T, Willms L. Audio-video fire-detection of open

fires[J]. Fire Safety Journal,2006,41(4);311-314.

23. Xie Y, Zhu J, Cao Y, et al. Efficient video fire detection exploiting

motion-flicker-based dynamic features and deep static features[J]. IEEE

Access, 2020, 8: 81904-81917.

24. Shahid M, Chien I, Sarapugdi W, et al. Deep spatial-temporal networks

for flame detection[J]. Multimedia Tools and Applications, 2021, 80(28):

35297-35318.

25. Zhang D, Xiao H, Wen Jie, Xu Y. Real-Time Fire Detection Method

with Multi-feature Fusion on YOLOv5. Pattern Recognition and Artificial

Intelligence, 2022, 35(6): 548-561.

26. Yuan J, Wang L, Wu P, et al. Detection of wildfires along transmission

lines using deep time and space features[J]. Pattern Recognition and Image

Analysis, 2018, 28(4): 805-812.

27. Wang Z, Wei D, Hu X. Research on two stage flame detection

algorithm based on fire feature and machine learning[C]//Proceedings of

the 2019 International Conference on Robotics, Intelligent Control and

Artificial Intelligence. 2019: 574-578.

28. Ryu J, Kwak D. Flame Detection Using Appearance-Based Pre-

Processing and Convolutional Neural Network[J]. Applied Sciences, 2021,

11(11): 5138.

29. Barnich O, Van Droogenbroeck M. ViBe: A universal background

subtraction algorithm for video sequences[J]. IEEE Transactions on Image

processing, 2010, 20(6): 1709-1724.

30. Shamsoshoara A, Afghah F, Razi A, et al. Aerial imagery pile burn

detection using deep learning: The FLAME dataset[J]. Computer

Networks, 2021, 193: 108001.

Author Name: Preparation of Papers for IEEE Access (February 2017)

8 VOLUME XX, 2017

31. Habiboğlu Y H, Günay O, Çetin A E. Covariance matrix-based fire

and flame detection method in video[J]. Machine Vision and Applications,

2012, 23(6): 1103-1113.

32. Oh S H, Ghyme S W, Jung S K, et al. Early wildfire detection using

convolutional neural network[C]//International Workshop on Frontiers of

Computer Vision. Springer, Singapore, 2020: 18-30.

33. Bouguettaya A, Zarzour H, Taberkit A M, et al. A review on early

wildfire detection from unmanned aerial vehicles using deep learning-

based computer vision algorithms[J]. Signal Processing, 2022, 190:

108309.

34. Rashkovetsky D, Mauracher F, Langer M, et al. Wildfire detection

from multisensor satellite imagery using deep semantic segmentation[J].

IEEE Journal of Selected Topics in Applied Earth Observations and

Remote Sensing, 2021, 14: 7001-7016.

35. Jindal P, Gupta H, Pachauri N, et al. Real-Time Wildfire Detection via

Image-Based Deep Learning Algorithm[C]//Soft Computing: Theories and

Applications: Proceedings of SoCTA 2020, Volume 2. Springer Singapore,

2021: 539-550.

36. Treneska S, Stojkoska B R. Wildfire detection from UAV collected

images using transfer learning[C]//Proceedings of the 18th International

Conference on Informatics and Information Technologies, Skopje, North

Macedonia. 2021: 6-7.

37. Abdelhamid A A, El-Kenawy E S M, Alotaibi B, et al. Robust speech

emotion recognition using CNN+ LSTM based on stochastic fractal search

optimization algorithm[J]. IEEE Access, 2022, 10: 49265-49284.

38. Lopac N, Hržić F, Vuksanović I P, et al. Detection of Non-Stationary

GW Signals in High Noise From Cohen ’ s Class of Time – Frequency

Representations Using Deep Learning[J]. IEEE Access, 2021, 10: 2408-

2428.

39. Ren S, He K, Girshick R, et al. Faster R-CNN: towards realtime object

detection with region proposal networks[J]. IEEE Transactions on Pattern

Analysis & Machine Intelligence, 2017, 6:1137-1149.

40. Liu W, Anguelov D, Erhan D, et al. SSD: Single shot multibox

detector[C]. Computer Vision – ECCV 2016: 14th European Conference,

Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14.

Springer International Publishing, 2016: 21-37.

LIDONG WANG was born in Wenzhou,

Zhejiang, China, in 1982. She received the Ph.D.

degree from the College of Computer Science

and Technology, Zhejiang University, in 2013.

She was a visiting scholar from Tongji University,

in 2022.

She is currently an Associate Professor with

Hangzhou Normal University. She has publised

more than 30 papers on social network analysis,

data mining, pattern recognition, and computer

education. Her current research interests include

image processing, machine learning, and text mining.

[1] World Health Organization. (Oct. 2004). WHO Policy Perspectives on Medicines. Looking at the Pharmacovigilance: Ensuring the Safe Use of Medicines. [Online]. Available: http://apps.who.int/medicinedocs/ pdf/s6164e/s6164e.pdf

[2] J. K. Aronson, ‘‘Artificial intelligence in pharmacovigilance: An introduction to terms, concepts, applications, and limitations,’’ Drug Saf., vol. 45, no. 5, pp. 407–418, May 2022, doi: 10.1007/s40264-022-01156-5.

[3] R. Ball and P. G. Dal, ‘‘‘Artificial intelligence’ for pharmacovigilance: Ready for prime time?’’ Drug Saf., vol. 45, no. 5, pp. 429–438, 2022.

[4] W. Samek, T. Wiegand, and K.-R. Müller, ‘‘Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models,’’ 2017, arXiv:1708.08296. [5] D. Gunning and D. Aha, ‘‘DARPA’s explainable artificial intelligence (XAI) program,’’ AI Mag., vol. 40, no. 2, pp. 44–58, 2019, doi: 10.1609/aimag.v40i2.2850.

[6] F. K. Došilovic, M. Brcic, and N. Hlupic, ‘‘Explainable artificial intelligence: A survey,’’ in Proc. 41st Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO), New York, NY, USA, May 2018, pp. 210–215.

[7] C. Meske and E. Bunde, ‘‘Transparency and trust in human-AI-interaction: The role of model-agnostic explanations in computer vision-based decision support,’’ in Proc. Int. Conf. Hum.-Comput. Interact. Berlin, Germany: Springer, Jul. 2020, pp. 54–69.

[8] A. Das and P. Rad, ‘‘Opportunities and challenges in explainable artificial intelligence (XAI): A survey,’’ 2020, arXiv:2006.11371.

[9] M. Tulio, S. Singh, and C. Guestrin, ‘‘Anchors: High-precision modelagnostic explanations,’’ in Proc. AAAI Conf. Artif. Intell., 2018, vol. 32, no. 1, pp. 1–15.

[10] I. R. Ward, L. Wang, J. Lu, M. Bennamoun, G. Dwivedi, and F. M. Sanfilippo, ‘‘Explainable artificial intelligence for pharmacovigilance: What features are important when predicting adverse outcomes?’’ Comput. Methods Programs Biomed., vol. 212, Nov. 2021, Art. no. 106415, doi: 10.1016/j.cmpb.2021.106415.

[11] Pharmacovigilance Indicators: A Practical Manual for the Assessment of Pharmacovigilance Systems, World Health Organization, Geneva, Switzerland, 2015.

[12] S. Zhang, S. Dev, J. Voyles, and A. S. Rao, ‘‘Attention-based multi-task learning in pharmacovigilance,’’ in Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM), Dec. 2018, pp. 22324–22328.

[13] J. Rebane, I. Samsten, P. Pantelidis, and P. Papapetrou, ‘‘Assessing the clinical validity of attention-based and SHAP temporal explanations for adverse drug event predictions,’’ in Proc. IEEE 34th Int. Symp. Comput.- Based Med. Syst. (CBMS), Jun. 2021, pp. 235–240. [14] E. Bresso, P. Monnin, C. Bousquet, F.-E. Calvier, N.-C. Ndiaye, N. Petitpain, M. Smaïl-Tabbone, and A. Coulet, ‘‘Investigating ADR mechanisms with explainable AI: A feasibility study with knowledge graph mining,’’ BMC Med. Informat. Decis. Making, vol. 21, no. 1, pp. 1–14, Dec. 2021.

[15] J. Wang, L.-C. Yu, and X. Zhang, ‘‘Explainable detection of adverse drug reaction with imbalanced data distribution,’’ PLOS Comput. Biol., vol. 18, no. 6, Jun. 2022, Art. no. e1010144.

[16] A. S. Mantripragada, S. P. Teja, R. R. Katasani, P. Joshi, and R. Ramesh, ‘‘Prediction of adverse drug reactions using drug convolutional neural networks,’’ J. Bioinf. Comput. Biol., vol. 19, no. 1, Feb. 2021, Art. no. 2050046, doi: 10.1142/S0219720020500468.

[17] P. Joshi, M. Vedhanayagam, and R. Ramesh, ‘‘An ensembled SVM based approach for predicting adverse drug reactions,’’ Current Bioinf., vol. 16, no. 3, pp. 422–432, Mar. 2021. [18] Z. Yu, H. Ji, J. Xiao, P. Wei, L. Song, T. Tang, X. Hao, J. Zhang, Q. Qi, Y. Zhou, F. Gao, and Y. Jia, ‘‘Predicting adverse drug events in Chinese pediatric inpatients with the associated risk factors: A machine learning study,’’ Frontiers Pharmacol., vol. 12, p. 516, Apr. 2021.

[19] Q. Wang, K. Huang, P. Chandak, M. Zitnik, and N. Gehlenborg, ‘‘Extending the nested model for user-centric XAI: A design study on GNNbased drug repurposing,’’ IEEE Trans. Vis. Comput. Graph., vol. 29, no. 1, pp. 1266–1276, Jan. 2023.

[20] Z. Li, Y. W. Lam, Q. Liu, A. Y. K. Lau, H. Y. Au-Yeung, and R. H. M. Chan, ‘‘Machine learning-driven drug discovery: Prediction of structure-cytotoxicity correlation leads to identification of potential antileukemia compounds,’’ in Proc. 42nd Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC), Jul. 2020, pp. 5464–5467. 50838 VOLUME 11, 2023 S. Lee et al.: XAI for Patient Safety: A Review of Application in Pharmacovigilance

[21] S. He, D. Zhao, Y. Ling, H. Cai, Y. Cai, J. Zhang, and L. Wang, ‘‘Machine learning enables accurate and rapid prediction of active molecules against breast cancer cells,’’ Frontiers Pharmacol., vol. 12, pp. 1–15, Dec. 2021, doi: 10.3389/fphar.2021.796534.

[22] J.-B. Lamy, K. Sedki, and R. Tsopra, ‘‘Explainable decision support through the learning and visualization of preferences from a formal ontology of antibiotic treatments,’’ J. Biomed. Informat., vol. 104, Apr. 2020,