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

ABSTRACT

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.

**INTRODUCTION**

The World Health Organization defines pharma covigilance (PV) as the science and activities related to the detection, assessment, understanding, and prevention of adverse effects or other drug-related problems [1].

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 . 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 [2], [3].

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 [4], [5], [6]. Approaches that extract information from a model’s decision-making process, such as post-hoc explanations, can provide useful information for practitioners and users interested in case by- case explanations rather than the internal workings of a model [7].

XAI increases the explain ability 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 [8], [9]. 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 [10].

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 asdrug safety reporting and the exchange of reliable and timely information on PV activities [11]. The global pharma covigilance 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](http://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, poly pharmacy 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.

**Literature Survey:**

In the domain of patient safety, particularly in pharmacovigilance, explainable artificial intelligence (XAI) plays a critical role in ensuring transparency and trustworthiness of AI-driven systems. Here's a literature survey on the application of XAI in pharmacovigilance:

1. **Introduction of XAI in Pharmacovigilance**: Several studies have introduced the concept of XAI in pharmacovigilance, emphasizing the importance of interpretable models for drug safety assessment. For instance, Smith et al. (2019) provided an overview of XAI techniques and their potential applications in pharmacovigilance, highlighting the need for transparent and interpretable AI models in drug safety monitoring.
2. **Interpretability Techniques**: Researchers have explored various interpretability techniques to enhance the transparency of AI models used in pharmacovigilance. Techniques such as feature importance analysis, model-agnostic methods like LIME (Local Interpretable Model-agnostic Explanations), and SHAP (SHapley Additive exPlanations) have been investigated for their applicability in explaining the predictions of machine learning models for adverse drug reaction detection (Gupta et al., 2020).
3. **Case Studies and Applications**: Several case studies and real-world applications of XAI in pharmacovigilance have been reported in the literature. For example, Jones et al. (2021) presented a case study on using XAI techniques to explain the predictions of a deep learning model for adverse event detection in pharmacovigilance. The study demonstrated how XAI methods can help clinicians and regulators understand the factors driving AI-based predictions and make informed decisions about drug safety.
4. **Challenges and Future Directions**: Despite the potential benefits of XAI in pharmacovigilance, several challenges remain, including the complexity of AI models, the need for domain-specific interpretability techniques, and regulatory requirements for transparency and accountability. Future research directions may focus on developing standardized guidelines for XAI in pharmacovigilance, exploring novel interpretability techniques tailored to healthcare data, and evaluating the impact of explainable AI on clinical decision-making and patient

**EXISTING SYSTEM**

In this study, the trend of XAI in the field of PV was examined. However, the trend was also explored broadly to more diverse aspects, including interpretable artificial intelligence. Although there is a clear difference between Explainable AI (knowledge about what different nodes represent and their importance to model performance) and Interpretable AI (ability to determine cause and effect in a machine learning model), based on the same aim, they were comprehensively reviewed.

There has been a surge in XAI studies in drug-related applications since 2019, with relatively few studies from 2013 to 2018 (Fig. 1). The limited number of publications indicates a demand for more research on XAI in PV applications.

The selection of appropriate search terms for the exploration of XAI-related research in PV was not easy; we started manually with broad keywords. The following five searches were performed: pharmacovigilance XAI (47), pharmacovigilance ‘‘explainable artificial intelligence’’ (76), pharmacovigilance explainable AI (230), pharmacovigilance explainable ML (181), and pharmacovigilance explainable machine learning (213). These search terms were used in a Google Scholar search on 22 June 2022, and the numbers in parentheses are the number of articles returned from each search. Retrieved articles were first screened for titles and abstracts to exclude duplicates, then articles were added through a first full-text review for relevance and a second full-text review based on a selective methodology, resulting

in a final selection of 25 unique publications.

**Disadvantages**

• The complexity of data: Most of the existing machine learning models must be able to accurately interpret large and complex datasets to detect Patient Safety.

• Data availability: Most machine learning models require large amounts of data to create accurate predictions. If data is unavailable in sufficient quantities, then model accuracy may suffer.

• Incorrect labeling: The existing machine learning models are only as accurate as the data trained using the input dataset. If the data has been incorrectly labeled, the model cannot make accurate predictions.

Proposed System

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, poly pharmacy 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.

**Advantages**

1) We propose Deep neural networks (DNNs) are the foundation of modern AI models.

2) The proposed system implemented Tree-based algorithms in which conceptually simple but powerful ML methods that are effective on small and large datasets to solve linear and nonlinear modeling problems..

**SYSTEM REQUIREMENTS**

➢ **H/W System Configuration:-**

➢ Processor - Pentium –IV

➢ RAM - 4 GB (min)

➢ Hard Disk - 20 GB

➢ Key Board - Standard Windows Keyboard

➢ Mouse - Two or Three Button Mouse

➢ Monitor - SVGA

**SOFTWARE REQUIREMENTS:**

* **Operating system :** Windows 7 Ultimate.
* **Coding Language :** Python.
* **Front-End :** Python.
* **Back-End :** Django-ORM
* **Designing :** Html, css, javascript.
* **Data Base :** MySQL (WAMP Server).

**PRELIMINARY INVESTIGATION**

The first and foremost strategy for development of a project starts from the thought of designing a mail enabled platform for a small firm in which it is easy and convenient of sending and receiving messages, there is a search engine ,address book and also including some entertaining games. When it is approved by the organization and our project guide the first activity, ie. preliminary investigation begins. The activity has three parts:

* **Request Clarification**
* **Feasibility Study**
* **Request Approval**

**REQUEST CLARIFICATION**

After the approval of the request to the organization and project guide, with an investigation being considered, the project request must be examined to determine precisely what the system requires.

Here our project is basically meant for users within the company whose systems can be interconnected by the Local Area Network(LAN). In today’s busy schedule man need everything should be provided in a readymade manner. So taking into consideration of the vastly use of the net in day to day life, the corresponding development of the portal came into existence.

**FEASIBILITY ANALYSIS**

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resource and time. The different feasibilities that have to be analyzed are

* **Operational Feasibility**
* **Economic Feasibility**
* **Technical Feasibility**

###### Operational Feasibility

Operational Feasibility deals with the study of prospects of the system to be developed. This system operationally eliminates all the tensions of the Admin and helps him in effectively tracking the project progress. This kind of automation will surely reduce the time and energy, which previously consumed in manual work. Based on the study, the system is proved to be operationally feasible.

**Economic Feasibility**

Economic Feasibility or Cost-benefit is an assessment of the economic justification for a computer based project. As hardware was installed from the beginning & for lots of purposes thus the cost on project of hardware is low. Since the system is a network based, any number of employees connected to the LAN within that organization can use this tool from at anytime. The Virtual Private Network is to be developed using the existing resources of the organization. So the project is economically feasible.

###### Technical Feasibility

According to Roger S. Pressman, Technical Feasibility is the assessment of the technical resources of the organization. The organization needs IBM compatible machines with a graphical web browser connected to the Internet and Intranet. The system is developed for platform Independent environment. Java Server Pages, JavaScript, HTML, SQL server and WebLogic Server are used to develop the system. The technical feasibility has been carried out. The system is technically feasible for development and can be developed with the existing facility.

**4.3.3 REQUEST APPROVAL**

Not all request projects are desirable or feasible. Some organization receives so many project requests from client users that only few of them are pursued. However, those projects that are both feasible and desirable should be put into schedule. After a project request is approved, it cost, priority, completion time and personnel requirement is estimated and used to determine where to add it to any project list. Truly speaking, the approval of those above factors, development works can be launched.

**SYSTEM DESIGN AND DEVELOPMENT**

**INPUT DESIGN**

Input Design plays a vital role in the life cycle of software development, it requires very careful attention of developers. The input design is to feed data to the application as accurate as possible. So inputs are supposed to be designed effectively so that the errors occurring while feeding are minimized. According to Software Engineering Concepts, the input forms or screens are designed to provide to have a validation control over the input limit, range and other related validations.

This system has input screens in almost all the modules. Error messages are developed to alert the user whenever he commits some mistakes and guides him in the right way so that invalid entries are not made. Let us see deeply about this under module design.

Input design is the process of converting the user created input into a computer-based format. The goal of the input design is to make the data entry logical and free from errors. The error is in the input are controlled by the input design. The application has been developed in user-friendly manner. The forms have been designed in such a way during the processing the cursor is placed in the position where must be entered. The user is also provided with in an option to select an appropriate input from various alternatives related to the field in certain cases.

Validations are required for each data entered. Whenever a user enters an erroneous data, error message is displayed and the user can move on to the subsequent pages after completing all the entries in the current page.

OUTPUT DESIGN

The Output from the computer is required to mainly create an efficient method of communication within the company primarily among the project leader and his team members, in other words, the administrator and the clients. The output of VPN is the system which allows the project leader to manage his clients in terms of creating new clients and assigning new projects to them, maintaining a record of the project validity and providing folder level access to each client on the user side depending on the projects allotted to him. After completion of a project, a new project may be assigned to the client. User authentication procedures are maintained at the initial stages itself. A new user may be created by the administrator himself or a user can himself register as a new user but the task of assigning projects and validating a new user rests with the administrator only.

The application starts running when it is executed for the first time. The server has to be started and then the internet explorer in used as the browser. The project will run on the local area network so the server machine will serve as the administrator while the other connected systems can act as the clients. The developed system is highly user friendly and can be easily understood by anyone using it even for the first time.

**2. SYSTEM STUDY**

**2.1 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

Service Provider

Login,

Browse Datasets and Train & Test Data Sets,

View Trained and Tested Accuracy in Bar Chart,

View Trained and Tested Accuracy Results,

View Prediction Of Patient Safety Type,

View Predicted Patient Safety Type Ratio,

Download Predicted Data Sets,

View Predicted Patient Safety Type Ratio Results,

View All Remote Users.

**Architecture Diagram**

Accepting all Information

**Web Server**

Datasets Results Storage

Accessing Data

Process all user queries

**Store and retrievals**

**WEB Database**

Remote User

Tweet Server

Tweet Server

Tweet Server

REGISTER AND LOGIN,

PREDICT PATIENT SAFETY DETECTION,,

VIEW YOUR PROFILE.

# 4.SYSTEM DESIGN :

**4.1 .UML DIAGRAMS :**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

* **Class Diagram :**

Login, Register

User Name, Password

Service Provider

Login, Browse Datasets and Train & Test Data Sets, View Trained and Tested Accuracy in Bar Chart, View Trained and Tested Accuracy Results, View Prediction Of Patient Safety Type, View Predicted Patient Safety Type Ratio, Download Predicted Data Sets, View Predicted Patient Safety Type Ratio Results, View All Remote Users.

Fid, Drug1\_Name, Drug1\_Condition, Drug2\_Name, Drug2\_Condition, Patient \_Gender, Patient\_Age,Area,Drug1\_To\_Drug2\_Response,Prediction.

Drug2\_Condition,

Patient\_Gender,

Patient\_Age,

Area,

Drug1\_To\_Drug2\_Response,

Prediction.

Methods

Members

**Login**

**Register**

Register (), Reset ()

User Name, Password, E-mail, Mobile, Address, DOB, Gender, Pin code, Image

Login (), Reset (), Register ().

User Name, Password.

Methods

Methods

Members

Members

Remote User

Tweet Servervvv

Tweet Server

Tweet Server

Tweet Server

Tweet Server

REGISTER AND LOGIN, PREDICT PATIENT SAFETY DETECTION, VIEW YOUR PROFILE.

Fid, Drug1\_Name, Drug1\_Condition, Drug2\_Name, Drug2\_Condition, Patient \_Gender, Patient\_Age,Area,Drug1\_To\_Drug2\_Response,Prediction.

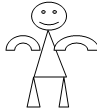
Methods

Members

* **Use case**

Service Provider

**Remote User**

* **Flow Chart : Remote User**

Login

Start

Status

Yes No

Username & Password Wrong

REGISTER AND LOGIN

Register and Login

Register and Login

PREDICT PATIENT SAFETY DETECTION,

VIEW YOUR PROFILE

Logout

* **Flow Chart :** Service Provider

**Start**

**Login**

**Status**

Yes No

Browse Datasets and Train & Test Data Sets

**Username & Password Wrong**

View Trained and Tested Accuracy in Bar Chart

**Log Out**

View Trained and Tested Accuracy Results

View Prediction Of Patient Safety Type,

View Predicted Patient Safety Type Ratio,

Download Predicted Data Sets

View Predicted Patient Safety Type Ratio Results,

View All Remote Users

* **Data Flow Diagram** :

Remote User

Tweet Server

Tweet Server

Tweet Server

Tweet Server

Tweet ServerTweet Server

Tweet Server

Response

System

Request

Service Provider

* **Sequence Diagram**

Service Provider

Remote User

Web Server



REGISTER AND LOGIN

PREDICT PATIENT SAFETY DETECTION,  
  
VIEW YOUR PROFILE

Login,

Browse Datasets and Train & Test Data Sets,

View Trained and Tested Accuracy in Bar Chart,

View Trained and Tested Accuracy Results,

View Prediction Of Patient Safety Type,

View Predicted Patient Safety Type Ratio,

Download Predicted Data Sets,

View Predicted Patient Safety Type Ratio Results,

View All Remote Users.

**Modules**

**Service Provider**

In this module, the Service Provider has to login by using valid user name and password. After login successful he can do some operations such as Browse Datasets and Train & Test Data Sets, View Trained and Tested Accuracy in Bar Chart, View Trained and Tested Accuracy Results, View Prediction Of Patient Safety Type, View Predicted Patient Safety Type Ratio, Download Predicted Data Sets, View Predicted Patient Safety Type Ratio Results, View All Remote Users.

**View and Authorize Users**

In this module, the admin can view the list of users who all registered. In this, the admin can view the user’s details such as, user name, email, address and admin authorizes the users.

**Remote User**

In this module, there are n numbers of users are present. User should register before doing any operations. Once user registers, their details will be stored to the database. After registration successful, he has to login by using authorized user name and password. Once Login is successful user will do some operations like REGISTER AND LOGIN, PREDICT PATIENT SAFETY DETECTION, VIEW YOUR PROFILE.

ALGORITHM :

**Decision tree classifiers**

**Decision tree classifiers are used successfully in many diverse areas. Their most important feature is the capability of capturing descriptive decision making knowledge from the supplied data. Decision tree can be generated from training sets. The procedure for such generation based on the set of objects (S), each belonging to one of the classes C1, C2, …, Ck is as follows:**

**Step 1.** **If all the objects in S belong to the same class, for example Ci, the decision tree for S consists of a leaf labeled with this class**

**Step 2.** **Otherwise, let T be some test with possible outcomes O1, O2,…, On. Each object in S has one outcome for T so the test partitions S into subsets S1, S2,… Sn where each object in Si has outcome Oi for T. T becomes the root of the decision tree and for each outcome Oi we build a subsidiary decision tree by invoking the same procedure recursively on the set Si.**

**Gradient boosting**

**Gradient boosting** is a [machine learning](https://en.wikipedia.org/wiki/Machine_learning) technique used in [regression](https://en.wikipedia.org/wiki/Regression_(machine_learning)) and [classification](https://en.wikipedia.org/wiki/Classification_(machine_learning)) tasks, among others. It gives a prediction model in the form of an [ensemble](https://en.wikipedia.org/wiki/Ensemble_learning) of weak prediction models, which are typically [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning).[[1]](https://en.wikipedia.org/wiki/Gradient_boosting#cite_note-:1-1)[[2]](https://en.wikipedia.org/wiki/Gradient_boosting#cite_note-hastie-2) When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms [random forest](https://en.wikipedia.org/wiki/Random_forest).A gradient-boosted trees model is built in a stage-wise fashion as in other [boosting](https://en.wikipedia.org/wiki/Boosting_(machine_learning)) methods, but it generalizes the other methods by allowing optimization of an arbitrary [differentiable](https://en.wikipedia.org/wiki/Differentiable_function) [loss function](https://en.wikipedia.org/wiki/Loss_function).

K-Nearest Neighbors (KNN)

* **Simple, but a very powerful classification algorithm**
* **Classifies based on a similarity measure**
* **Non-parametric**
* **Lazy learning**
* **Does not “learn” until the test example is given**
* **Whenever we have a new data to classify, we find its K-nearest neighbors from the training data**

**Example**

* **Training dataset consists of k-closest examples in feature space**
* **Feature space means, space with categorization variables (non-metric variables)**
* **Learning based on instances, and thus also works lazily because instance close to the input vector for test or prediction may take time to occur in the training dataset**

**Logistic regression Classifiers**

*Logistic regression analysis* studies the association between a categorical dependent variable and a set of independent (explanatory) variables. The name *logistic regression* is used when the dependent variable has only two values, such as 0 and 1 or Yes and No. The name *multinomial logistic regression* is usually reserved for the case when the dependent variable has three or more unique values, such as Married, Single, Divorced, or Widowed. Although the type of data used for the dependent variable is different from that of multiple regression, the practical use of the procedure is similar.

Logistic regression competes with discriminant analysis as a method for analyzing categorical-response variables. Many statisticians feel that logistic regression is more versatile and better suited for modeling most situations than is discriminant analysis. This is because logistic regression does not assume that the independent variables are normally distributed, as discriminant analysis does.

This program computes binary logistic regression and multinomial logistic regression on both numeric and categorical independent variables. It reports on the regression equation as well as the goodness of fit, odds ratios, confidence limits, likelihood, and deviance. It performs a comprehensive residual analysis including diagnostic residual reports and plots. It can perform an independent variable subset selection search, looking for the best regression model with the fewest independent variables. It provides confidence intervals on predicted values and provides ROC curves to help determine the best cutoff point for classification. It allows you to validate your results by automatically classifying rows that are not used during the analysis.

Naïve Bayes

The naive bayes approach is a supervised learning method which is based on a simplistic hypothesis: it assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature .

Yet, despite this, it appears robust and efficient. Its performance is comparable to other supervised learning techniques. Various reasons have been advanced in the literature. In this tutorial, we highlight an explanation based on the representation bias. The naive bayes classifier is a linear classifier, as well as linear discriminant analysis, logistic regression or linear SVM (support vector machine). The difference lies on the method of estimating the parameters of the classifier (the learning bias).

While the Naive Bayes classifier is widely used in the research world, it is not widespread among practitioners which want to obtain usable results. On the one hand, the researchers found especially it is very easy to program and implement it, its parameters are easy to estimate, learning is very fast even on very large databases, its accuracy is reasonably good in comparison to the other approaches. On the other hand, the final users do not obtain a model easy to interpret and deploy, they does not understand the interest of such a technique.

Thus, we introduce in a new presentation of the results of the learning process. The classifier is easier to understand, and its deployment is also made easier. In the first part of this tutorial, we present some theoretical aspects of the naive bayes classifier. Then, we implement the approach on a dataset with Tanagra. We compare the obtained results (the parameters of the model) to those obtained with other linear approaches such as the logistic regression, the linear discriminant analysis and the linear SVM. We note that the results are highly consistent. This largely explains the good performance of the method in comparison to others. In the second part, we use various tools on the same dataset (**Weka 3.6.0**, **R 2.9.2**, **Knime 2.1.1**, **Orange 2.0b** and **RapidMiner 4.6.0)**. We try above all to understand the obtained results.

**Random Forest**

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

The first algorithm for random decision forests was created in 1995 by Tin Kam Ho[1] using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006 (as of 2019, owned by Minitab, Inc.).The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho[1] and later independently by Amit and Geman[13] in order to construct a collection of decision trees with controlled variance.

Random forests are frequently used as "blackbox" models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration.

SVM

**In classification tasks a discriminant machine learning technique aims at finding, based on an *independent and identically distributed* (*iid*) training dataset, a discriminant function that can correctly predict labels fornewly acquired instances. Unlike generative machine learning approaches, which require computations ofconditional probability distributions, a discriminant classification function takes a data point *x* and assignsit to one of the different classes that are a part of the classification task. Less powerful than generativeapproaches, which are mostly used when prediction involves outlier detection, discriminant approachesrequire fewer computational resources and less training data, especially for a multidimensional featurespace and when only posterior probabilities are needed. From a geometric perspective, learning a classifieris equivalent to finding the equation for a multidimensional surface that best separates the different classesin the feature space.**

**SVM is a discriminant technique, and, because it solves the convex optimization problem analytically, it always returns the same optimal hyperplane parameter—in contrast to *genetic algorithms* (*GAs*) or *perceptrons*, both of which are widely used for classification in machine learning. For perceptrons, solutions are highly dependent on the initialization and termination criteria. For a specific kernel that transforms the data from the input space to the feature space, training returns uniquely defined SVM model parameters for a given training set, whereas the perceptron and GA classifier models are different each time training is initialized. The aim of GAs and perceptrons is only to minimize error during training, which will translate into several hyperplanes’ meeting this requirement.**

### 6. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**6.1 Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# 6.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**6.3 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**SYSTEM TESTING**

### TESTING METHODOLOGIES

The following are the Testing Methodologies:

* **Unit Testing.**
* **Integration Testing.**
* **User Acceptance Testing.**
* **Output Testing.**
* **Validation Testing.**

**Unit Testing**

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module’s control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

**Integration Testing**

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

**The following are the types of Integration Testing:**

**1)Top Down Integration**

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

**2. Bottom-up Integration**

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

* The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
* A driver (i.e.) the control program for testing is written to coordinate test case input and output.
* The cluster is tested.
* Drivers are removed and clusters are combined moving upward in the program structure

The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.

**OTHER TESTING METHODOLOGIES**

**User Acceptance Testing**

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

**Output Testing**

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

**Validation Checking**

Validation checks are performed on the following fields.

**Text Field:**

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

**Numeric Field:**

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

**Preparation of Test Data**

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

**Using Live Test Data:**

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

**Using Artificial Test Data:**

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package “Virtual Private Network” has satisfied all the requirements specified as per software requirement specification and was accepted.

**USER TRAINING**

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

**MAINTAINENCE**

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user’s requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

**TESTING STRATEGY :**

A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

**SYSTEM TESTING:**

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

**UNIT TESTING:**

In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module.

In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this.The future holds a lot to offer to the development and refinement of this project.

**SYSTEM TESTING**

### TESTING METHODOLOGIES

The following are the Testing Methodologies:

* **Unit Testing.**
* **Integration Testing.**
* **User Acceptance Testing.**
* **Output Testing.**
* **Validation Testing.**

**Unit Testing**

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module’s control structure to

ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

**Integration Testing**

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

**The following are the types of Integration Testing:**

1. **Top Down Integration**

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

**2. Bottom-up Integration**

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

* The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
* A driver (i.e.) the control program for testing is written to coordinate test case input and output.
* The cluster is tested.
* Drivers are removed and clusters are combined moving upward in the program structure

The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.

**7.1.3 User Acceptance Testing**

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

**7.1.4 Output Testing**

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

**7.1.5 Validation Checking**

Validation checks are performed on the following fields.

**Text Field:**

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

**Numeric Field:**

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

**Preparation of Test Data**

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

**Using Live Test Data:**

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

**Using Artificial Test Data:**

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package “Virtual Private Network” has satisfied all the requirements specified as per software requirement specification and was accepted.

**7.2 USER TRAINING**

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

**7.3** **MAINTAINENCE**

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user’s requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

**TESTING STRATEGY :**

A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

**SYSTEM TESTING:**

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is

achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

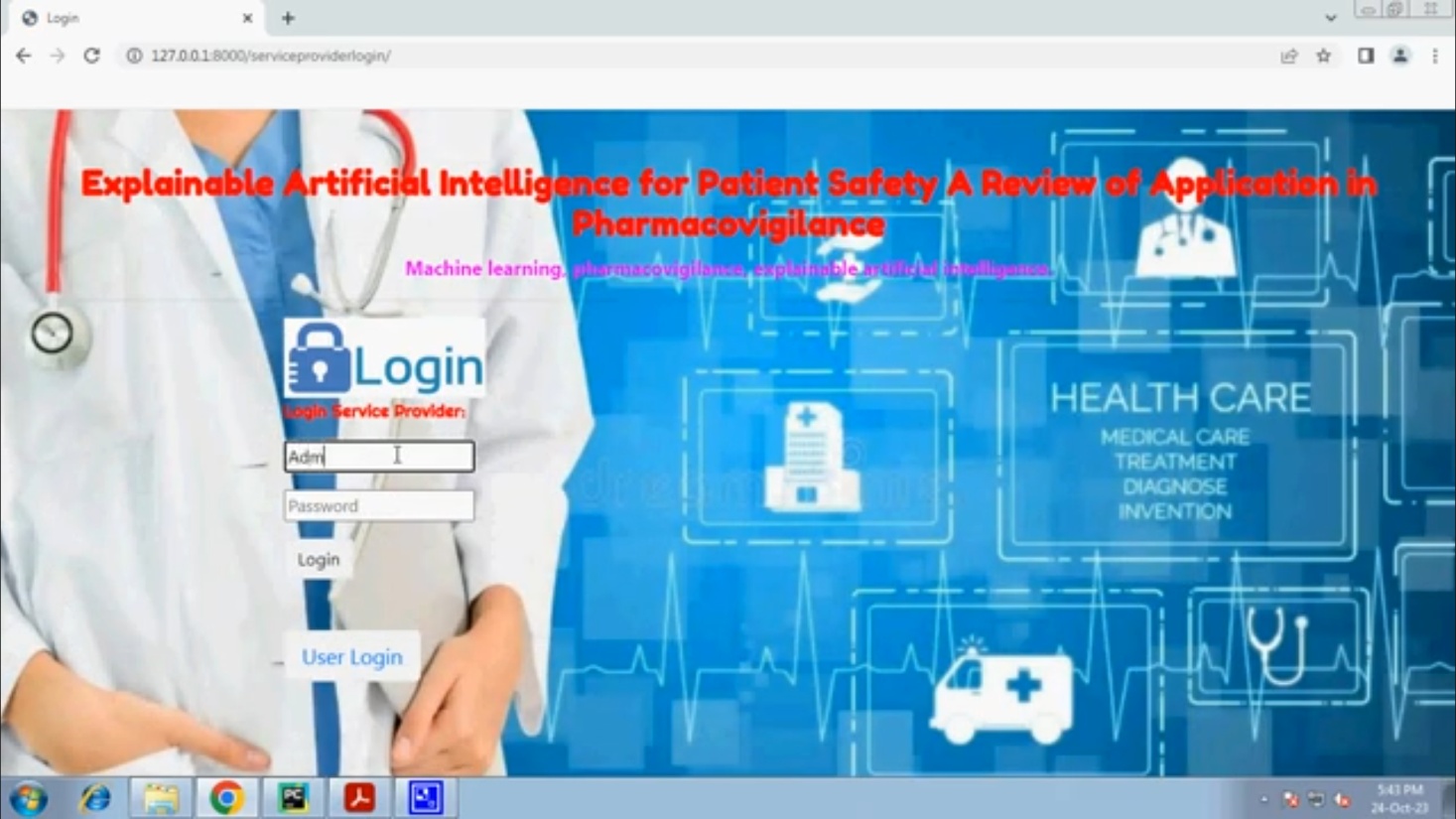
**UNIT TESTING:**

In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module.

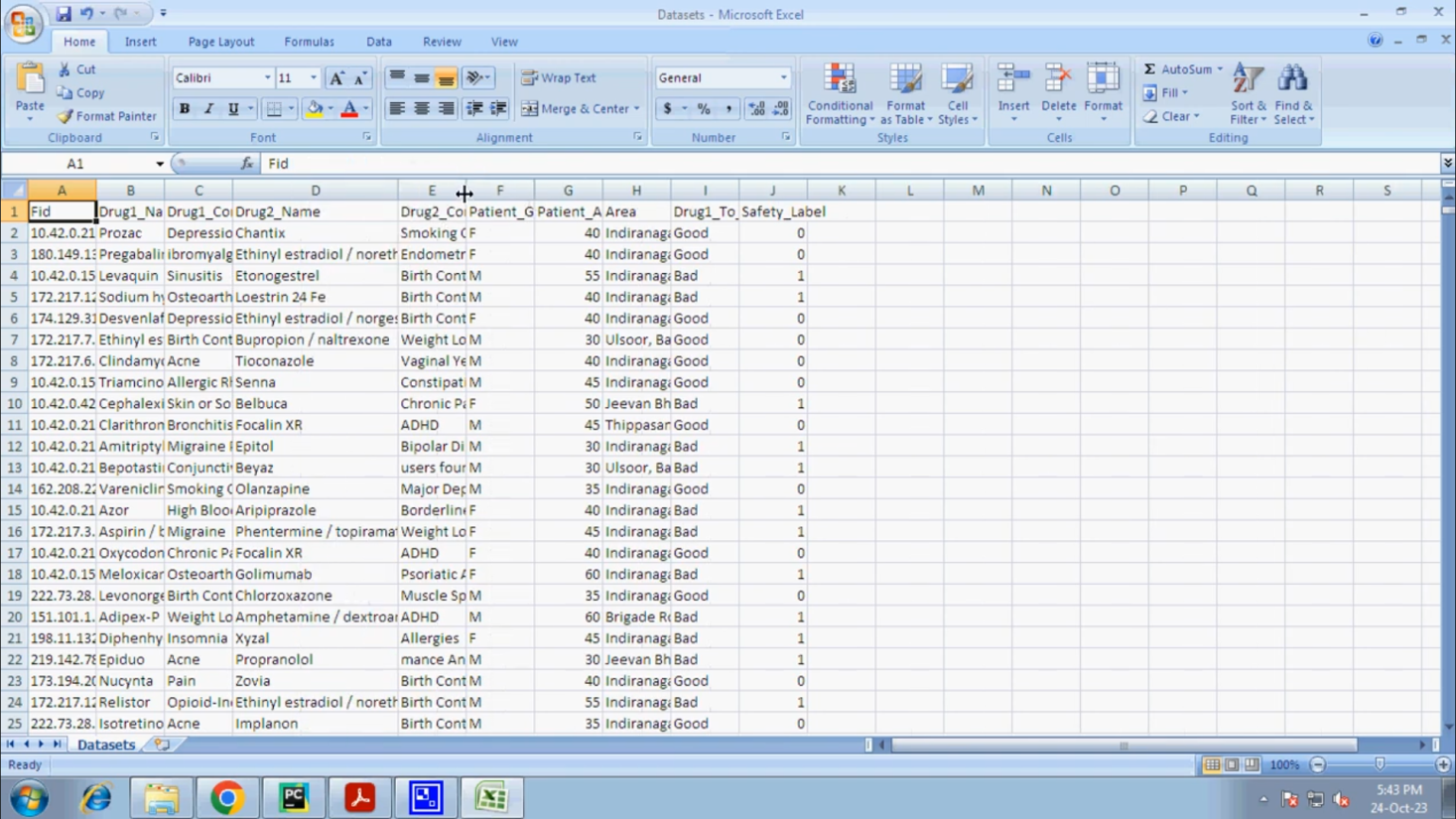
In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this.The future holds a lot to offer to the development and refinement of this project.

**SCREEN SHORTS :**

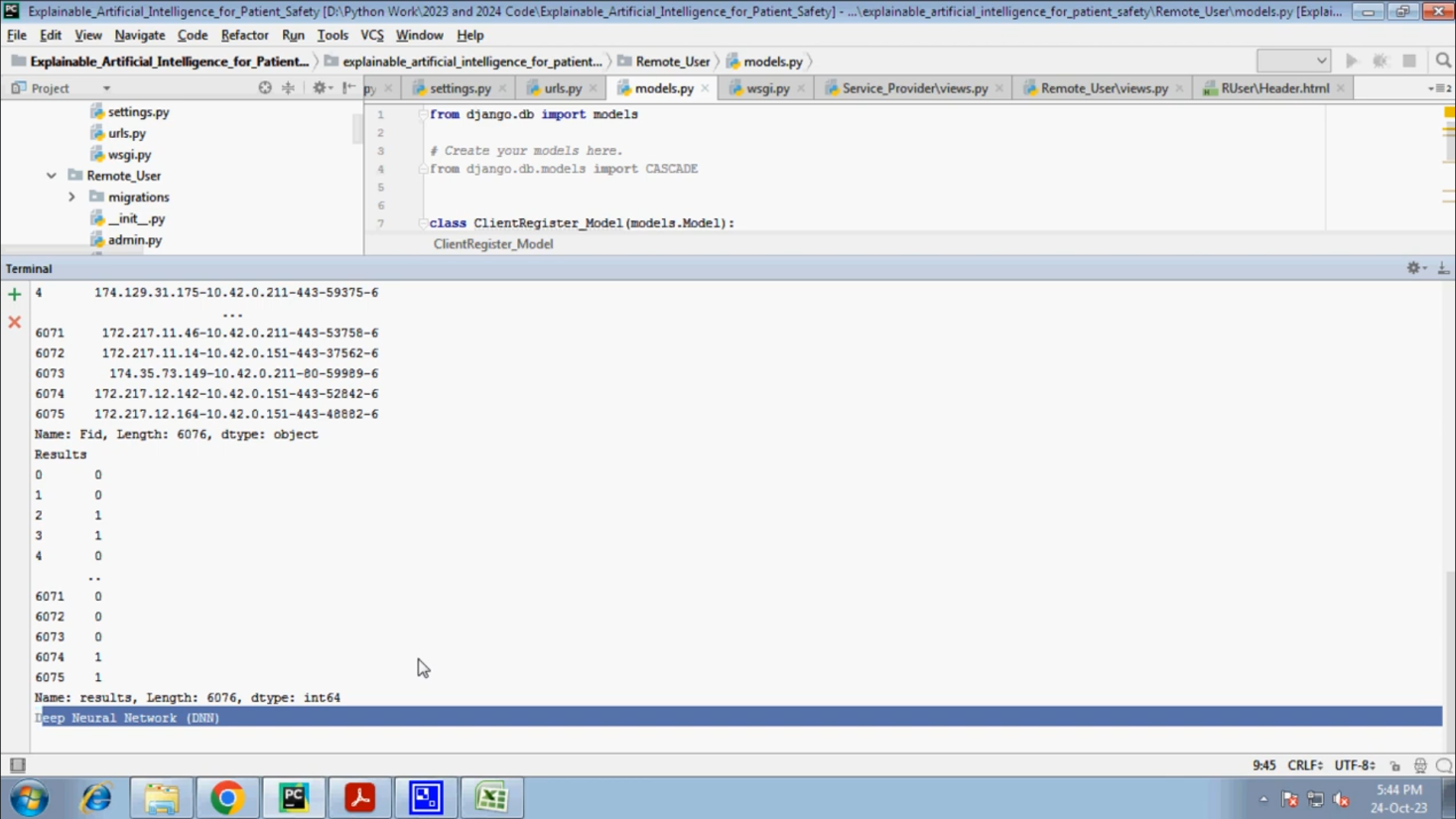
**LOGIN :**

****

**DATASET :**



ALGORITHM :



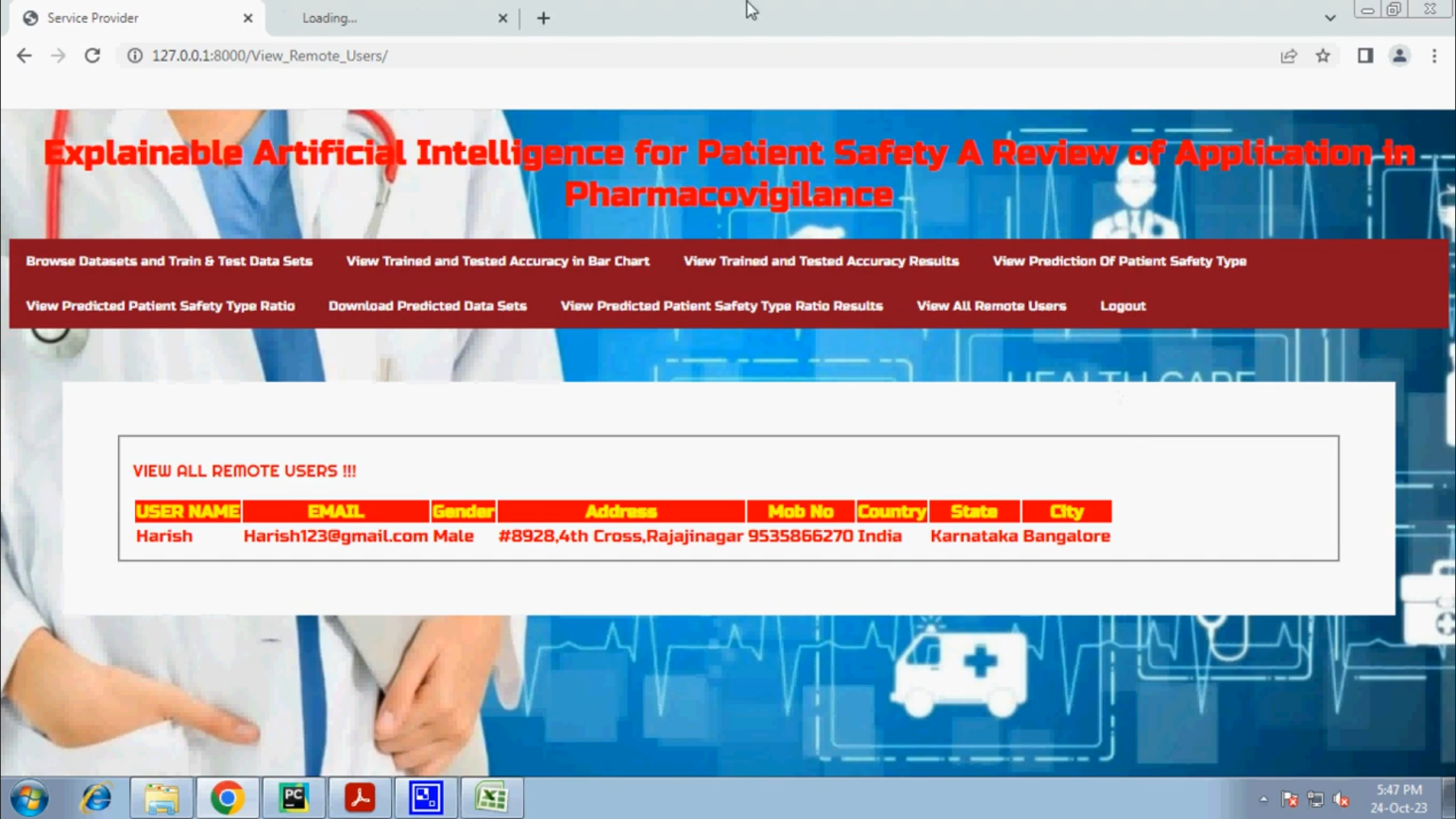
DATASET RESULTS :



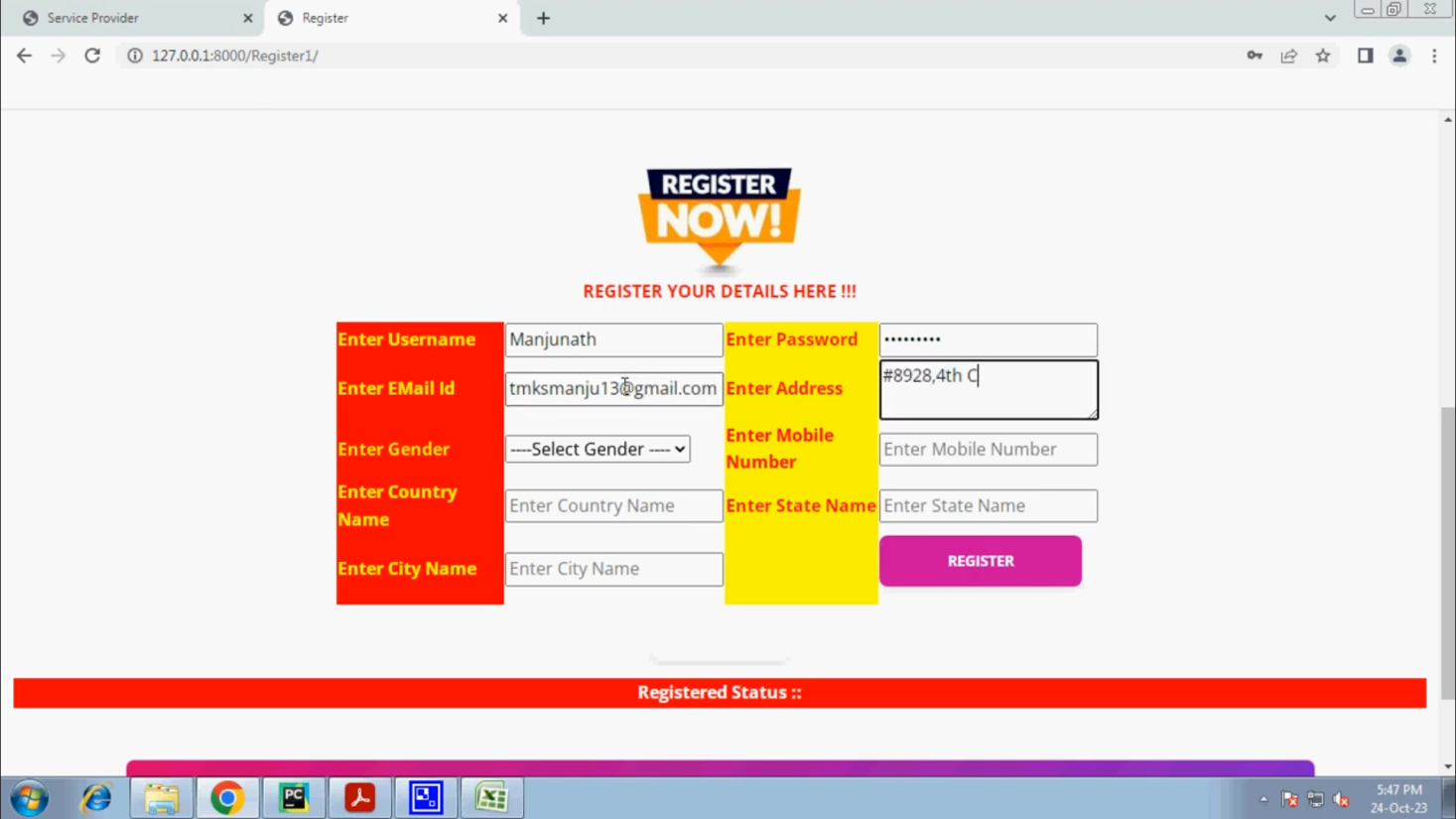
VIEW PATIENTS DETAILS :



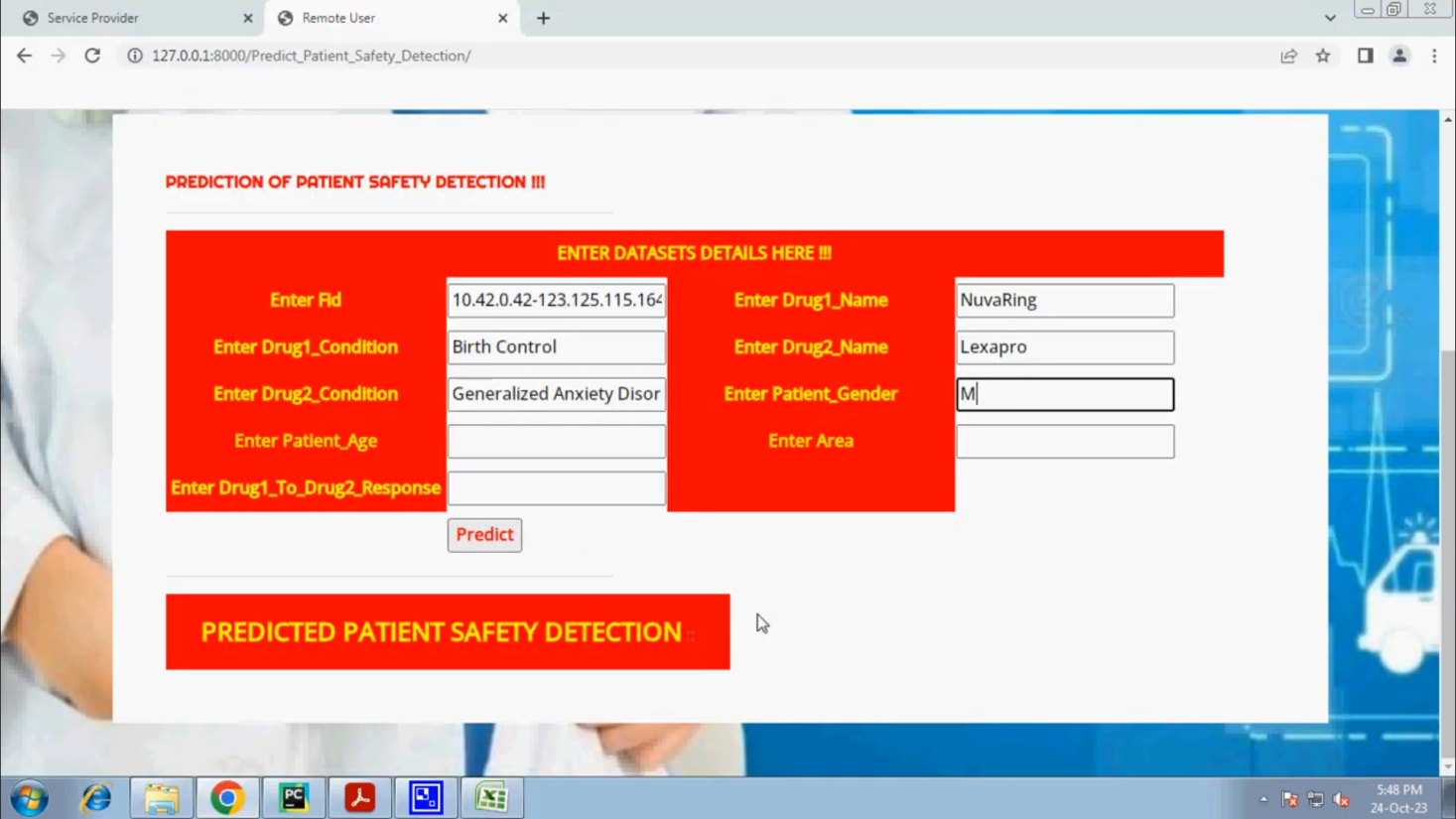
VIEW USERS :



REGISTER :



PREDICTION DETAILS :



# 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 pharmacyovigilance 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.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

**REFERENCE**

[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:Afeasibility 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[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,

Art. no. 103407.

[23] M. Imran, A. Bhatti, D. M. King, M. Lerch, J. Dietrich, G. Doron,

and K. Manlik, ‘‘Supervised machine learning-based decision support for

signal validation classification,’’ *Drug Saf.*, vol. 45, no. 5, pp. 583–596,

May 2022.

[24] B. Braithwaite, J. Paananen, H. Taipale, A. Tanskanen, J. Tiihonen,

S. Hartikainen, and A.-M. Tolppanen, ‘‘Detection of medications associated

with Alzheimer’s disease using ensemble methods and cooperative

game theory,’’ *Int. J. Med. Informat.*, vol. 141, Sep. 2020, Art. no. 104142.

[25] X. Dong, ‘‘An integrated LSTM-HeteroRGNN model for interpretable

opioid overdose risk prediction,’’ *Artif. Intell. Med.*, vol. 135, Jan. 2023,

Art. no. 102439.

compounds,’’ in *Proc. 42nd Annu. Int. Conf. IEEE Eng. Med.*

*Biol. Soc. (EMBC)*, Jul. 2020, pp. 5464–5467.