A Project Stage-II Report on

ENHANCING DRUG SIDE EFFECTS PREDICTION WITH EXPLAINABLE AI FOR MEDICAL HEALTH APPLICATIONS

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

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (DATASCIENCE)

By

RAMPOLA BHARATH REDDY (21UJ1A6731)
VEMUGANTI SUSHANTH REDDY (21UJ1A6739)
NIRUGONDA SUSHMA (22UJ5A6709)
CHINNI SAI KRISHNA (22UJ5A6706)

Under the esteemed guidance o

Mr.MOHAMMAD YOUNUS M.Tech,

Assistant Professor

Department of Computer Science and Engineering (Data Science)

At

MALLA REDDY ENGINEERING COLLEGE AND MANAGEMENT SCIENCES

Kistapur (V), Medchal (M), Medchal Malkajgiri (Dist)-501401.

(An UGC autonomous Institution, NBA Accredited in CSE, ECE, IT, EEE)

2021-2025

Affiliated to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
KUKATPALLY, HYDERABAD-85.

MALLA REDDY ENGINEERING COLLEGE AND MANAGEMENT SCIENCES

Kistapur (V), Medchal (M), Medchal Malkajgiri (Dist)-501401.

(An UGC autonomous Institution, NBA Accredited in CSE, ECE, IT, EEE)

CERTIFICATE

This is to certify that the Project Stage-II report entitled "**ENHANCING DRUG SIDE EFFECTS PREDICTION WITH EXPLAINABLE AI FOR MEDICAL HEALTH APPLICATIONS** "is the bonafide work carried out and submitted by

RAMPOLA BHARATH REDDY	**(21UJ1A6731)**
VEMUGANTI SUSHANTH REDDY	**(21UJ1A6739)**
NIRUGONDA SUSHMA	**(22UJ5A6709)**
CHINNI SAI KRISHNA	**(22UJ5A6706)**

To the department of **Computer Science and Engineering (Data Science), Malla Reddy Engineering College and Management Sciences**, in partial fulfilment for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)** during the academic year 2024-2025.

Internal Guide
Mr.MOHAMMAD YOUNUS M.Tech,
Assistant Professor
Dept .of CSE (Data Science)

Head of The Department
Dr.Zaheer Sultana M.Tech, PhD.
Assistant Professor
Dept .of CSE (Data Science)

EXTERNAL

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that we have now the opportunity to express my guidance for all of them.

Our thankful to **Mr.V.MALLA REDDY** chairman of Malla Reddy Engineering College and Management Sciences for accepting ours and providing us with an opportunity to do a project in their esteemed organization.

Our thankful to Principal **Dr.M.SREEDHAR REDDY** M.Tech., Ph.D. Malla Reddy Engineering College and Management Sciences for helped ours to undergo project work as a part of university curriculum.

Our special thanks to **Dr. ZAHEER SULTANA** M.Tech,PhD.. Assistant Professor& Head in Computer Science and Engineering (Data Science) Department and **Mr.MOHAMMAD YOUNUS** M.Tech, Assistant Professor in Computer Science and Engineering (Data Science) Department for guiding us in the right way to complete our project in the right time.

We would like to thank our internal project mates and department faculties for their full-fledged guidance and giving courage to carry out the project.

Our very much thankful to one and all that helped us for the successful completion of our project.

`

RAMPOLA BHARATH REDDY (21UJ1A6731)
VEMUGANTI SUSHANTH REDDY (21UJ1A6739)
NIRUGONDA SUSHMA (22UJ5A6709)
CHINNI SAI KRISHNA (22UJ5A6706)

ABSTRACT

Drugs are medications that change how the body functions. Some make us feel good when ill, while others treat ongoing health issues. They are in the form of pills, syrups, creams, or injections. Examples: paracetamol, antibiotics, vitamins, allergy medications, blood pressure medication, diabetes medications, and cough syrups. Side effects of drugs are able to result in serious health hazards for patients. Before using a drug, it is important to be aware of the negative effects Machine learning models scan drug and patient information to identify patterns. The doctors are able to understand and trust AI advice because of these explanations. Important elements that could cause negative effects are identified by the system.

Adverse drug reactions (ADRs) are harmful effects that can occur when taking medicines, and they are a serious concern in the medical field. In this project, our aim to predict drug side effects using Artificial Intelligence (AI) with a focus on Explainable AI (XAI) techniques. We use different types of drug data like chemical structure and side effect records—and apply machine learning models to find patterns. To make these models understandable, which explain how the AI is making decisions. This helps doctors trust the system and make safer choices for patients. Our project supports better healthcare by improving drug safety and decision-making through smart and explainable AI tools.

.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABBREVATIONS

DR-SED	Drug Recommendation and Side Effects Detection
NLP	Natural Language Processing
ML	Machine Learning
MAE	Medication Administration Errors
TF-IDF	Terms Frequency Inverse Document Frequency
TF	Terms Frequency
MLP	Multilayer Perceptron
UML	Unified Modeling Language
IDLE	Integrated Development and Learning Environment
SVC	Support Vector Classifier
SGDC	Stochastic gradient descent classifier
EHR	Electronic Health Records
GUI	Graphical User Interface

CHAPTER 1

INTRODUCTION

1.1 Overview

A Drug Recommendation and side Effect Detection (DR-SED) system represents a cutting-edge technological solution within the healthcare domain, aiming to assist healthcare professionals in making informed decisions about medication selection and dosage. At its core, the system utilizes a combination of advanced algorithms, data analytics, and medical knowledge to analyze patient information and recommend the most appropriate medications based on individual needs and clinical circumstances. By leveraging vast repositories of medical literature, electronic health records (EHRs), and patient demographics, these systems can provide tailored recommendations that account for factors such as medical history, comorbidities, drug interactions, and patient preferences. This personalized approach marks a departure from traditional prescribing practices, which often rely on generalized guidelines and may not fully consider the nuances of each patient's condition.

The development of DR-SED systems has been fueled by several factors, including the increasing complexity of healthcare, the proliferation of medical information, and the need for precision medicine. As medical knowledge continues to expand at an exponential rate, healthcare professionals are faced with the daunting task of staying abreast of the latest research findings, drug interactions, and treatment guidelines. In this context, DR-SED systems serve as invaluable tools for synthesizing vast amounts of data and distilling actionable insights that can inform clinical decision-making. Moreover, these systems hold the potential to enhance patient safety by minimizing medication errors, adverse drug reactions, and drug-drug interactions, thereby improving overall healthcare quality and outcomes.

1.2 History

The history of DR-SED systems utilizing Natural Language Processing (NLP) framework traces its roots back to the early 2000s when the intersection of healthcare and computational linguistics began to gain momentum. Initially, early attempts at developing such systems were limited by the availability of data, computational resources, and the sophistication of NLP algorithms. However, pioneering research initiatives laid the foundation for subsequent advancements by demonstrating the feasibility of using NLP techniques to analyze medical texts, extract relevant information, and generate actionable insights related to drug therapy and patient care. These early endeavors set the

stage for the emergence of more robust and scalable NLP-driven recommendation systems in the years to come.

As computational power increased and NLP algorithms became more sophisticated, researchers and healthcare practitioners began to explore the potential applications of NLP in DR-SED systems. The proliferation of electronic health records (EHRs) and digital healthcare platforms further catalyzed this trend, providing access to vast repositories of structured and unstructured medical data that could be leveraged to enhance clinical decision-making processes. Over time, advancements in machine learning, natural language understanding, and semantic analysis enabled the development of NLP frameworks capable of parsing complex medical texts, identifying relevant clinical concepts, and generating personalized DR-SEDs tailored to individual patient profiles and treatment objectives.

The evolution of NLP-driven DR-SED systems has been shaped by ongoing research efforts, technological innovations, and real-world applications across diverse healthcare settings. From early prototype systems to contemporary implementations deployed in clinical practice, the trajectory of these systems reflects a continuous refinement of methodologies, algorithms, and data integration strategies. Moreover, as healthcare stakeholders increasingly recognize the potential benefits of NLP-driven recommendation systems in optimizing medication therapy, improving patient outcomes, and reducing healthcare costs, investment in research and development in this field has surged. Looking ahead, the history of DR-SED systems using NLP framework underscores the transformative impact of interdisciplinary collaboration between computer science, medicine, and data analytics in advancing the frontiers of healthcare innovation and improving the quality of patient care.

1.3 Problem Statement

The problem statement surrounding DR-SED systems using Natural Language Processing (NLP) framework encompasses several key challenges that impede optimal medication therapy and patient care. First and foremost is the issue of information overload within the healthcare domain. With the exponential growth of medical literature, drug databases, and patientrecords, healthcare providers are inundated with vast amounts of heterogeneous data that are often difficult to interpret and integrate into clinical decision-making processes. Navigating this sea of information manually is time-consuming and error-prone, leading to suboptimal treatment outcomes, medication errors, and patient safety concerns.

Another critical challenge is the variability and complexity of patient data. Medical records contain a wealth of information, including patient demographics, medical history, laboratory

results, and clinical notes, which are often recorded in unstructured or semi-structured formats. Extracting relevant insights from these heterogeneous data sources and synthesizing them into actionable recommendations requires sophisticated NLP techniques capable of understanding medical terminology, identifying clinical concepts, and contextualizing patient-specific factors. Moreover, the dynamic nature of healthcare data presents additional challenges related to data quality, consistency, and interoperability, which must be addressed to ensure the accuracy and reliability of DR-SED systems.

1.4 Research Motivation

The research motivation behind DR-SED systems using Natural Language Processing (NLP) framework is driven by several compelling factors within the healthcare landscape. Firstly, there exists a pressing need to optimize medication therapy and enhance patient outcomes. Traditional approaches to drug prescription often rely on standardized guidelines and may not fully account for individual patient characteristics, medical histories, and treatment preferences. By leveraging NLP techniques to analyze unstructured medical text, such as clinical notes and patient records, recommendation systems can generate personalized medication suggestions tailored to specific patient profiles, thereby improving treatment efficacy, minimizing adverse drug reactions, and promoting patient safety.

Moreover, the escalating complexity of healthcare delivery and the proliferation of medical knowledge underscore the importance of leveraging technology to augment clinical decision-making processes. As medical literature expands and treatment options evolve, healthcare providers face the daunting challenge of staying abreast of the latest developments and integrating this knowledge into practice. NLP-driven DR-SED systems offer a scalable and efficient means of synthesizing vast amounts of medical information, identifying relevant clinical insights, and disseminating evidence-based recommendations to healthcare professionals in real-time. By empowering providers with actionable intelligence, these systems have the potential to streamline healthcare workflows, reduce cognitive load, and facilitate data-driven decision-making at the point of care.

1.5 Application

- Clinical Decision Support: Helps doctors identify potential side effects before prescribing medications.
- Drug Safety Monitoring: Improves systems that track and report drug side effects,

  allowing for faster responses to safety concerns.

- Personalized Treatment: Assists in creating tailored treatment plans based on how individual patients might react to medications.
- Drug Development: Aids pharmaceutical companies in assessing potential side effects early in the drug development process.
- Patient Education: Provides information to patients about possible side effects, helping them make informed choices about their medications.
- Healthcare Policy: Informs regulators and policymakers about drug safety, leading to better guidelines for medication use.
- Research: Supports researchers in studying new drug interactions and side effects, enhancing the understanding of how drugs work.

CHAPTER 2

LITERATURE REVIEW

[1] J. Ramos (2003) presented a foundational study on the use of TF-IDF (Term Frequency–Inverse Document Frequency) to evaluate word relevance in document queries. His work explained how TF-IDF helps to quantify the importance of words within documents, which is especially valuable in text mining and information retrieval tasks. The method reduces the weight of common terms while highlighting unique, meaningful words relevant to specific queries. This technique became a standard for feature extraction in natural language processing and has been widely applied in biomedical text analysis, including drug-related literature.

[2] Shimada et. al developed a decision support system that helps doctors select appropriate first-line drugs. The system classifies patients' abilities to protect themselves from infectious diseases as a risk level for infection. In an evaluation of the prototype system, the risk level it determined correlated with the decisions of specialists. The system is very effective and convenient for doctors to use.

[3] He et. al presented a novel adaptive synthetic (ADASYN) sampling approach for learning from imbalanced data sets. The essential idea of ADASYN is to use a weighted distribution for different minority class examples according to their level of difficulty in learning, where more synthetic data is generated for minority class examples that are harder to learn compared to those minority examples that are easier to learn.

[4] Lei et. al presented a novel approach to polarity classification of short text snippets, which takes into account the way data are naturally distributed into several topics in order to obtain better classification models for polarity. This approach is multi-step, where in the initial step a standard topic classifier is learned from the data and the topic labels, and in the ensuing step several polarity classifiers, one per topic, are learned from the data and the polarity labels. They empirically show that our approach improves classification accuracy over a real-world dataset by over 10%, when compared against a standard single-step approach using the same feature sets. The approach is applicable whenever training material is available for building both topic and polarity learning models.

[5] Nikfarjam and Gonzalez et. al presented a new method for using association rules for colloquial text mining. They applied our method on user comments to find mentions of adverse reactions to drugs by extracting frequent patterns. Since we are dealing with highly informal

colloquial text, the idea of using extracted patterns might, at first, seem counter-intuitive. However, we indeed found consistencies in the user comments. This evaluation measured the effectiveness of this technique in extracting frequent patterns in this context. However, this method can easily be generalized for other contexts and languages.

[6] Doulaverakis et. al presented a DR-SED system based on Semantic Web technologies, termed GalenOW. It has been shown that OWL and Semantic Web technologies can provide a good match for DR-SEDs as OWL is expressive enough to effectively encapsulate medical knowledge. Rule-based reasoning can model medical decision making and aid experts. A comparison of the semantic-enabled implementation to a traditional business logic implementation was presented. Although the latter has shown better performance in time and memory requirements, semantic technologies provide a better alternative for integrating knowledge in the system than simple rule engines.

[7] Goeuriot et. al presented creation of lexical resources and their adaptation to the medical domain. We first describe the creation of a general lexicon, containing opinion words from the general domain and their polarity. Then they presented the creation of a medical opinion lexicon, based on a corpus of drug reviews. They show that some words have a different polarity in the general domain and in the medical one. Some words considered generally as neutral are opinionated in medical texts. They finally evaluate the lexicons and show with a simple algorithm that using our general lexicon gives better results than other well-known ones on our corpus and that adding the domain lexicon improves them as well.

[8] Keers et. al appraised empirical evidence relating to the causes of medication administration errors (MAEs) in hospital settings. Limited evidence from studies included in this systematic review suggests that MAEs are influenced by multiple systems factors, but if and how these arise and interconnect to lead to errors remains to be fully determined. Further theoretical focused is needed to investigate the MAE causation pathway, with an emphasis on ensuring interventions designed to minimise MAEs target recognised underlying causes of errors to maximise their impact.

[9]Wittich et. al provides a practicing physicians that focuses on medication error terminology and definitions, incidence, risk factors, avoidance strategies, and disclosure and legal consequences. A medication error is any error that occurs at any point in the medication use process. It has been estimated by the Institute of Medicine that medication errors cause 1 of 131 outpatient and 1 of 854 inpatient deaths. Medication factors (eg, similar sounding names, low therapeutic index), patient factors (eg, poor renal or hepatic function, impaired cognition,

polypharmacy), and health care professional factors (eg, use of abbreviations in prescriptions and other communications, cognitive biases) can precipitate medication errors.

[10]Zhang et. al proposed a novel cloud-assisted DR-SED (CADRE), which can recommend users with top-N related medicines according to symptoms. In CADRE, they first cluster the drugs into several groups according to the functional description information, and design a basic personalized DR-SED based on user collaborative filtering. Then, considering the shortcomings of collaborative filtering algorithm, such as computing expensive, cold start, and data sparsity, they propose a cloud-assisted approach for enriching end-user Quality of Experience (QoE) of DR-SED, by modeling and representing the relationship of the user, symptom and medicine via tensor decomposition. Finally, the proposed approach is evaluated with experimental study based on a real dataset crawled from Internet.

[11] Danushka et. al proposed an unsupervised method for learning domain-specific word representations that accurately capture the domain-specific aspects of word semantics. First, we select a subset of frequent words that occur in both domains as \emph{pivots}. Next, they optimize an objective function that enforces two constraints: for both source and target domain documents, pivots that appear in a document must accurately predict the co-occurring non-pivots, and, word representations learnt for pivots must be similar in the two domains. Moreover, they propose a method to perform domain adaptation using the learnt word representations. This proposed method significantly outperforms competitive baselines including the state-of-the-art domain-insensitive word representations, and reports best sentiment classification accuracies for all domain-pairs in a benchmark dataset.

[12] Sarker et. al suggested that interest in the utilization of the vast amounts of available social media data for ADR monitoring is increasing. In terms of sources, both health-related and general social media data have been used for ADR detection—while health-related sources tend to contain higher proportions of relevant data, the volume of data from general social media websites is significantly higher. There is still very limited amount of annotated data publicly available , and, as indicated by the promising results obtained by recent supervised learning approaches, there is a strong need to make such data available to the work community.

[13] Nikfarjam et. al introduced ADRMine, a machine learning-based concept extraction system that uses conditional random fields (CRFs). ADRMine utilizes a variety of features, including a novel feature for modeling words' semantic similarities. The similarities are modeled by clustering words based on unsupervised, pretrained word representation vectors (embeddings) generated from unlabeled user posts in social media using a deep learning technique.

[14] Tekade and Emmanuel et. al used Modeling Based on Probabilistic Approach a more fine-grained aspect level opinion mining. It is interesting to apply the model to find aspects relating to different segmentation of data such as different age groups or other attributes. It is also interesting to work with aspect interpretation as aspects are now represented by a list of keywords. If a few sentences can be extracted or generated automatically to summarize the keywords, interpretation & understanding will be improved.

[15] Sun et. al aimed at exploiting the rich information in doctor orders and developing data-driven approaches for improving clinical treatments. To this end, they first propose a novel method to measure the similarities between treatment records with consideration of sequential and multifaceted information in doctor orders. Then, they propose an efficient density-based clustering algorithm to summarize large-scale treatment records, and extract a semantic representation of each treatment cluster. Finally, they develop a unified framework to evaluate the discovered treatment regimens, and find the most effective treatment regimen for new patients. In the empirical study, they validate this methods with EMRs of 27,678 patients from 14 hospitals.

[16] Li et. al regard hashtag recommendation as a classification task but proposed a novel recurrent neural network model to learn vector-based tweet representations to recommend hashtags. More precisely, they use a skip-gram model to generate distributed word representations and then apply a convolutional neural network to learn semantic sentence vectors. Afterwards, they make use of the sentence vectors to train a long short-term memory recurrent neural network (LSTM-RNN). They directly use the produced tweet vectors as features to classify hashtags without any feature engineering. Experiments on real world data from Twitter to recommend hashtags show that our proposed LSTM-RNN model outperforms state-of-the-art methods and LSTM unit also obtains the best performance compared to standard RNN and gated recurrent unit (GRU).

[17] Korkontzelos et. al shows that adding sentiment analysis features can marginally improve the performance of even a state-of-the-art ADR identification method. This improvement can be of use to pharmacovigilance practice, due to the rapidly increasing popularity of social media and health forums.

[18] Gopalakrishnan et. al This work aims to apply neural network-based methods for opinion mining from social web in health care domain. We have extracted the reviews of two different drugs. Experimental analysis is done to analyze the performance of classification methods on reviews of two different drugs. The results demonstrate that neural network-based opinion mining approach outperforms the support vector machine method in terms of precision, recall and f-score.

It is also shown that the performance of radial basis function neural network method is superior to probabilistic neural network method in terms of the performance measures used.

[19] Alireza et. al aimed to identify ways to prevent medication errors in the hospital wards in Iran. This qualitative content analysis study was conducted on 16 nurses and 1 physician from 10 August 2019 to 30 March 2020. The participants were selected using a purposive sampling method. Data were collected using semi-structured interviews. Thematic analysis was used to extract key themes from the data.

[20] Gurdin et. al generated freely available datasets of WebMD.com drug reviews and star ratings for Common, Cancer, Depression, Diabetes, and Hypertension drugs. We explored four supervised learning models: Naive Bayes, Random Forests, Support Vector Machines, and Convolutional Neural Networks for the purpose of determining the polarity of drug reviews. We conducted inter-domain and cross-domain evaluations. They found that SVM obtained the highest f-measure on average and that cross-domain training produced similar or higher results to models trained directly on their respective datasets.

[21]Garg et. al presented a drug recommender system that can drastically reduce specialists heap. In this research, we build a medicine recommendation system that uses patient reviews to predict the sentiment using various vectorization processes like Bow, TF-IDF, Word2Vec, and Manual Feature Analysis, which can help recommend the top drug for a given disease by different classification algorithms. The predicted sentiments were evaluated by precision, recall, f1score, accuracy, and AUC score. The results show that classifier LinearSVC using TF-IDF vectorization outperforms all other models with 93% accuracy.

[22]Das P., Mazumder D.H. This paper offers a comprehensive survey of supervised machine learning techniques used over the past 20 years for predicting drug side effects. It discusses various models like decision trees, SVM, and neural networks. The study emphasizes improvements in model performance and practical applications in pharmacovigilance. It also highlights data sources, challenges, and research gaps in ADR prediction.

[23]Downing N.S. et al. The authors explore postmarketing safety issues for new drugs approved by the FDA between 2001 and 2010. They found that many therapeutics face adverse safety events after approval. This paper highlights the urgent need for predictive safety surveillance tools. It supports the integration of AI to mitigate risks in early drug use phases.

[24] Craveiro S.N. et al. (2020) This review analyzes cases of drug withdrawal due to safety concerns and how decisions are made. The findings indicate that ADRs are a primary cause of

withdrawal. The study suggests improving pre-market testing and post-market monitoring. It reinforces the need for predictive modeling to prevent harmful drugs from reaching patients.

[25] Lavertu A. et al. (2021) This paper introduces the concept of digital pharmacovigilance and the transition toward real-world data monitoring. It emphasizes AI's role in identifying ADRs using electronic records and social media. The authors propose a future framework for real-time ADR detection. The study marks a shift from reactive to proactive safety assessment.

[26] Vora L.K. et al. (2023) This study explores the impact of artificial intelligence on pharmaceutical technologies and drug delivery systems. It demonstrates how ML improves drug design, release kinetics, and formulation. AI is shown to reduce toxicity through data-driven simulations. The paper underlines AI’s contribution to personalized dosage and smart delivery.

[27] Yang S., Kar S. (2023) The paper focuses on using AI and machine learning for early detection of adverse drug reactions (ADRs). It discusses using clinical data and NLP to detect drug-induced toxicity. The authors demonstrate how predictive models can reduce healthcare burdens. It supports the deployment of ML in pharmacovigilance pipelines.

[28] Biala G. et al. (2023) This research emphasizes the application of modern AI techniques in drug discovery and toxicity prevention. The study investigates how machine learning and neural networks can optimize compound selection. It highlights reduced ADR risks during the early development phase. The integration of AI into traditional pharmacology is well-supported.

[29] Han R. et al. (2023) The authors demonstrate how artificial intelligence transforms early drug discovery in medicinal chemistry. AI tools are used to analyze molecular interactions and predict harmful effects. The study presents several models that identify compounds with reduced toxicity. It positions AI as a vital tool in minimizing side effects.

[30] Johnson K.B. et al. (2021) This paper discusses how AI supports precision medicine to customize treatment plans. The integration of clinical, genetic, and behavioral data enhances patient outcomes. AI-driven platforms predict adverse responses and improve drug matching. The study envisions safer and more effective healthcare with intelligent systems.

[31] Singh S. et al. (2023) This work bridges the gap between pharmacological research and data analytics using AI. It emphasizes the need for ML in drug side effect prediction and repurposing. Various algorithms and data fusion methods are discussed. The paper presents AI as essential for modern pharmaceutical research.

[32] Alowais S.A. et al. (2023) The paper reviews AI's integration in clinical practice for improving diagnosis and treatment. It explores intelligent systems that assist physicians in risk assessment. The research supports the use of AI for predicting drug side effects in patient-specific contexts. It promotes AI as a core healthcare technology.

[33] Sachdev K., Gupta M.K. (2020) This review summarizes computational methods for predicting side effects of drugs. It covers ML algorithms, molecular fingerprints, and pharmacogenomic data analysis. The study provides a comparative view of techniques and tools. It recommends hybrid systems for better prediction accuracy.

[34] Ho T.B. et al. (2016) This paper introduces a data-driven model for ADR prediction using patient records and prescription histories. It employs clustering and classification to detect adverse events. Results show improved prediction accuracy over conventional methods. The research supports integrating data mining with clinical systems.

[35] Deimazar G., Sheikhtaheri A. (2023) A systematic review of ML models to predict patient safety events using electronic health records (EHRs). The paper evaluates algorithms based on accuracy, interpretability, and integration ease. EHR-based ML models show potential in real-time ADR detection. The study emphasizes ethical use and data privacy.

[36] Rajpoot K. et al. (2022) This chapter outlines various in silico methods for drug toxicity prediction. It includes QSAR modeling, docking simulations, and toxicophore identification. These methods help in early rejection of harmful compounds. The research enhances safety in pre-clinical drug development stages.

[37] Liu M. et al. (2012) A large-scale study predicting ADRs using chemical, biological, and phenotypic features of drugs. It integrates multi-modal data into ML models for ADR classification. The model achieved good performance across several metrics. It demonstrates the power of cross-domain data fusion.

[38] Pauwels E. et al. (2011) Proposes a chemical fragment-based approach for predicting drug side effects. The method correlates structural components with known ADR profiles. It shows that side effects can be anticipated before full compound synthesis. This technique offers faster and safer drug screening.

[39] Mizutani S. et al. (2012) Relates drug-protein interaction networks to observed side effects. The study uses network analysis to map potential ADR pathways. It suggests that drugs affecting

similar proteins may cause similar side effects. The approach supports mechanism-level side effect prediction.

[40] Amaro R.E., Mulholland A.J. (2018) This review explores multiscale simulations to address complexity in drug design. It combines molecular and system-level data to predict therapeutic effects. The approach improves understanding of drug behavior in biological systems. It is useful in predicting unintended interactions.

[41] Duran-Frigola M., Aloy P. (2013) Analyzes chemical and biological properties to understand mechanisms behind drug side effects. The authors use feature integration to identify high-risk drugs. The study supports using AI to uncover ADR mechanisms. This aids in preemptive drug screening.

[42] Boland M.R. et al. (2016) Proposes systems biology methods for ADR identification. It integrates genomics, transcriptomics, and proteomics to understand drug response. The study supports the development of multi-omics AI models. These approaches enhance biological interpretability of side effects.

[43] Yoo S. et al. (2018) Introduces an in silico system for systemic drug effect profiling. It predicts potential interactions and side effects using simulations. The study supports early detection of ADRs using virtual screening. It enhances patient safety by minimizing trial-and-error.

[44] Zitnik M. et al. (2019) Reviews machine learning techniques for integrating diverse biological datasets. It presents a framework for combining genomics, imaging, and drug data. The goal is to improve prediction of drug responses and risks. The study advocates for interpretable AI in healthcare.

[45] Marques L. et al. (2024) Reviews in silico innovations in precision medicine and ADR prediction. It includes AI-powered simulation tools and patient-specific modeling. The paper presents use cases in drug personalization and risk reduction. It promotes virtual trials for future drug testing.

[46] Arksey H., O'Malley L. (2005) This foundational paper introduces a methodological framework for conducting scoping reviews. It guides systematic mapping of literature, particularly in health sciences. The framework is widely used in AI-healthcare reviews. It ensures comprehensive and unbiased evidence collection.

[47] Chen T. et al. (2023) Presents ADR prediction using the SIDER dataset and machine learning techniques. It emphasizes feature importance analysis to interpret ML outputs. The model identifies key drug attributes responsible for side effects. The paper also supports the use of explainable AI.

[48] Wu Z., Chen L. (2022)
This paper proposes a similarity-based method that integrates multiple feature sampling to predict drug side effects. By combining structural, chemical, and biological similarities, the model enhances ADR prediction accuracy. The method effectively handles the high-dimensional feature space. It shows promising results on benchmark datasets in biomedical informatics.

[49] Güneş S.S. et al. (2021) The study focuses on in silico prediction of adverse reactions specifically for antidepressant drugs. It employs computational toxicology and machine learning to reduce trial-and-error in psychiatric drug prescriptions. The model simulates drug behavior to forecast possible side effects. The authors emphasize "Primum non nocere" – first, do no harm.

[50] Zhou H.Y. et al. (2020) This paper introduces MEDICASCY, a machine learning platform that predicts drug side effects, indications, and mechanisms of action. It utilizes small-molecule features and biomedical data to classify drugs. The model performs multi-task learning to increase its generalizability. It shows high precision in multiple drug analysis tasks.

[51] Seo S. et al. (2020) The authors use comprehensive similarity measures—chemical, phenotypic, and target-based—to improve side effect prediction. Their method integrates multiple sources to enhance the learning process. It demonstrates strong prediction performance using public drug datasets. The paper highlights the importance of multimodal fusion in pharmacovigilance.

[52] Jiang H. et al. (2020) This research moves from empirical risk to structural risk minimization for predicting drug side-effect profiles. It incorporates feature selection, regularization, and model complexity control. The proposed model improves generalizability while maintaining accuracy. The work is significant in avoiding overfitting in biomedical predictive systems.

[53] Galeano D. et al. (2020) This study predicts the *frequency* of drug side effects rather than just their presence, offering a novel angle in ADR research. Using advanced graph-based machine learning, the model quantifies the likelihood of various side effects. It allows better prioritization in pharmacovigilance systems. The results highlight the importance of frequency modeling in drug safety.

[54] Afdhal D. et al. (2020) The authors propose a method using multi-label linear discriminant analysis combined with multi-label learning to predict multiple ADRs simultaneously. Their model captures label dependencies effectively, offering improved ADR prediction for complex drugs. The approach is evaluated on publicly available datasets. It demonstrates potential in large-scale multi-reaction detection.

[55] Muñoz E. et al. (2019) This paper uses knowledge graphs along with multi-label learning models for ADR prediction. The model leverages structured biomedical relationships between drugs, proteins, and phenotypes. It improves interpretability and supports explainable AI in drug safety. The integration of semantic networks with ML offers enhanced decision support tools.

[56] Jamal S. et al. (2019) Focused on cardiovascular drug reactions, this study designs computational models for adverse outcome prediction. It evaluates molecular features and patient history data to forecast risks. The approach supports early detection of harmful cardiovascular effects. It underscores the importance of domain-specific ADR modeling.

[57] Zhao X. et al. (2018) The paper proposes a similarity-based method using heterogeneous data sources such as drug properties, targets, and side-effect profiles. The fusion of diverse data boosts ADR prediction performance. It applies matrix factorization and graph techniques. The study shows robustness in predicting complex drug reaction

[58] Zheng Y. et al. (2017) This conference paper presents an optimized drug similarity framework to enhance side-effect prediction accuracy. It incorporates a variety of similarity metrics including structural, target, and ATC code-based similarity. The optimization technique improves model adaptability across different drug types. The method achieves high efficiency in identifying potential ADRs.

[59] Sun C. et al. (2017) The authors develop a comprehensive drug similarity-based model to predict drug side effects. The framework fuses information from multiple domains including chemical structure and pharmacological properties. It demonstrates improved accuracy compared to traditional single-source models. The study emphasizes the benefits of holistic similarity analysis.

[60] Niu Y.Q., Zhang W. (2017) This work introduces a quantitative prediction model that utilizes diverse drug-related features for side effect prediction. The study applies machine learning algorithms to model complex feature relationships. Results show improved recall and precision for ADR detection. It supports a feature-centric view in computational drug safety.

[61] Guoliang Tan et al. (2025) The authors propose SMVSNN, a deep learning framework using Spiking Multi-view Siamese Neural Networks to predict anticancer drug–drug interactions. It captures temporal and structural information from multiple data views. The model is especially tailored to handle sparse and high-dimensional biological datasets. Its high accuracy demonstrates suitability for oncology drug analysis.

[62] Gökhan Tahıl et al. (2024) This study critiques how stereoisomers challenge ML models in drug discovery and ADR prediction. It reveals the limitations of current feature extraction techniques when processing stereochemical diversity. The findings call for improved representations in cheminformatics. The work offers important insights into model limitations in pharmacology.

[63] Kenneth M. Merz et al. (2024) This editorial discusses the role of machine learning in bio-cheminformatics, focusing on its integration with biological and chemical datasets. It highlights key challenges such as data sparsity, feature selection, and model interpretability. The authors advocate for hybrid modeling approaches and cross-disciplinary collaboration. This work sets the stage for future innovations in ADR prediction using ML.

[64] Bin Yang et al. (2025) The study applies graph attention networks (GATs) to predict drug–drug interactions, particularly in traditional Chinese medicine. It constructs knowledge graphs to represent complex compound relationships and their pharmacological effects. The model enhances interpretability and predictive accuracy. This is a notable step toward applying GNNs in herbal-based drug safety research.

[65] Liuxi Chu et al. (2025) This work presents a hydrogel-based delivery system co-administering FGF21 and H_2S for diabetic wound healing. While not solely focused on ADRs, it contributes by optimizing spatiotemporal drug delivery to reduce toxicity. The study supports AI-guided therapeutic design for controlled release. It shows how smart biomaterials can complement computational predictions.

[66] Linqian Zhao et al. (2025) An adaptive multi-kernel graph neural network (GNN) is introduced for drug–drug interaction prediction. The model dynamically selects the best kernel to capture different feature relationships. Its adaptability improves generalization across varying biomedical contexts. The study highlights the future of flexible GNN frameworks in pharmacovigilance.

[67] Linqian Zhao et al. (2025) This paper proposes a mutual-guided co-attention mechanism with a heterogeneous graph-based framework for drug interaction event prediction. It improves upon standard GNNs by emphasizing context-aware learning. The model captures drug attributes more effectively and enhances interaction classification. This contributes to better ADR event forecasting in complex medical settings.

[68] Zengqian Deng et al. (2025) The authors introduce MAVGAE, a multimodal framework using variational graph autoencoders for predicting asymmetric drug–drug interactions. It combines structural, semantic, and relational data to capture the nuances in drug behavior. The model significantly improves prediction of non-reciprocal interactions. Its design supports more accurate identification of unexpected ADRs.

[69] Jiahui Zhang et al. (2025) This study proposes IIB-DDI, a model based on invariant information bottleneck theory to address out-of-distribution drug–drug interaction prediction. It aims to generalize predictions beyond training data by retaining only the most essential features. The method ensures robustness in novel drug scenarios. It offers a solution for real-world application challenges in pharmacovigilance.

[70] Xiao-Hui Yue et al. (2025) This research evaluates an AI-based prescription decision system deployed in primary healthcare settings. The system uses evidence-based auditing to minimize prescription errors and adverse effects. It leads to safer and more effective medication practices. The paper underscores AI’s potential to transform routine clinical workflows and reduce ADR incidents.

[71] Abhayjit Singh Gulati et al. (2025) An ensemble-based machine learning model is proposed for predicting drug–drug interactions. By combining multiple classifiers, the system improves reliability and reduces prediction variance. It performs well across several ADR-related datasets. The ensemble strategy enhances robustness, especially in noisy or imbalanced datasets.

[72] Ting Zhang et al. (2025) The study presents CA-SQBG, a cross-attention Siamese quantum BiGRU model for extracting drug–drug interaction events. It leverages quantum computation principles alongside deep learning. The model excels in understanding relational patterns in unstructured data. It showcases cutting-edge AI applications in biomedical text mining.

[73] Marios Spanakis et al. (2025) reviewed artificial intelligence models and tools focused on assessing drug–herb interactions. They highlighted how AI enhances the identification and prediction of potential adverse interactions between herbal supplements and pharmaceutical drugs.

Their study emphasized the importance of integrating these AI tools in clinical practice to prevent harmful side effects and improve patient safety.

[74] Yingbo Zhang et al. (2025) conducted a comprehensive survey evaluating ChatGPT's capabilities in pharmacology. They assessed how large language models align with human expertise in drug-related queries and decision-making. The survey illustrated the potential of AI in supporting pharmacological research, education, and clinical applications by improving information accessibility and accuracy.

[75] Terry Adirim (2025) discussed the current and emerging uses of generative AI in clinical care, administration, and research. The study showed how generative AI tools streamline medical documentation, assist in clinical decision-making, and accelerate research processes. The author stressed the transformative impact of AI in healthcare workflows and patient management.

[76] Mithun Bhowmick et al. (2025) explored the future prospects of AI in drug discovery. They examined how AI accelerates the identification of novel drug candidates and optimizes the drug development pipeline. Their work outlined AI's potential to reduce time, cost, and failure rates in bringing new therapeutics to market.

[77] Ayman Mohamed Mostafa et al. (2025) proposed an innovative method combining tailored semantic embedding with machine learning to precisely predict the seriousness of drug-drug interactions. Their approach improved prediction accuracy and provided interpretable insights into interaction risks, helping healthcare providers make informed decisions.

[78] Ziyang Wang et al. (2025) reviewed the impact of food physical properties on oral drug absorption. The comprehensive analysis highlighted how food characteristics such as viscosity and particle size affect drug bioavailability. Understanding these interactions aids in designing better drug delivery systems to enhance therapeutic efficacy.

[79] Yilan Li et al. (2025) utilized AI-assisted hypothesis generation with ChatGPT and GPT-4o to tackle challenges in cardiotoxicity research. Their simulation study demonstrated how AI can support the identification of novel hypotheses, accelerating research into drug-induced heart toxicity and improving drug safety evaluations.

[80] Shambo Samrat Samajdar et al. (2025) reviewed current trends and future directions of artificial intelligence in healthcare. They discussed AI's role in diagnostics, personalized

treatment, and health management systems. The paper highlighted challenges like data privacy and emphasized the need for ethical AI adoption to maximize healthcare benefits.

CHAPTER 3

EXISTING SYSTEM

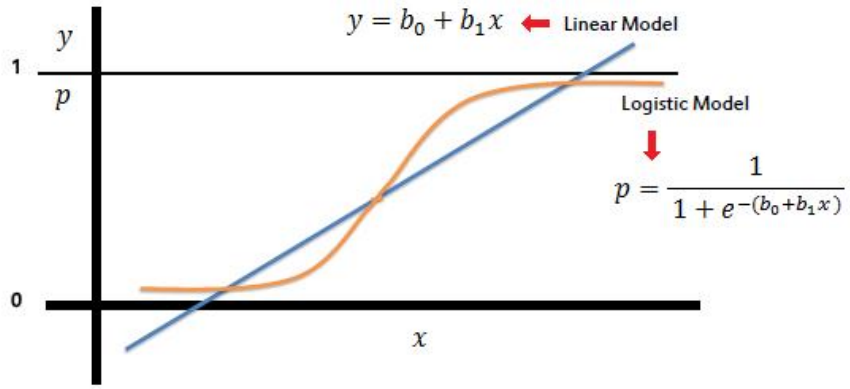
Recommender frameworks point to supply clients with personalized stock and repair to alter the expanding online information over-burden drawback. Various recommender framework methods are anticipated since the mid1990s, and numerous shapes of recommender framework code were created as of late for a spread of applications. The health-related substance shared through on-line feedbacks or surveys contains covered up assumption designs that emerges through totally distinctive sources from medical world which offer benefits to the pharmaceutical industry. Amid this, the on-line component is fantastically standard of late for online looking, diverse stock through distinctive websites like on-line buying of drugs at entryway step. Numerous websites and blogs offers clients to rate their stock with their fulfillment and quality of stock, logistics, administrations and criticism etc., which the clients examines for a particular medicine or on quality of administration

3.1 Logistic Regression

Logistic regression predicts the probability of an outcome that can only have two values (i.e. a dichotomy). The prediction is based on the use of one or several predictors (numerical and categorical). A linear regression is not appropriate for predicting the value of a binary variable for two reasons:

- A linear regression will predict values outside the acceptable range (e.g. predicting probabilities
- outside the range 0 to 1)
- Since the dichotomous experiments can only have one of two possible values for each experiment, the residuals will not be normally distributed about the predicted line.

On the other hand, a logistic regression produces a logistic curve, which is limited to values between 0 and 1. Logistic regression is similar to a linear regression, but the curve is constructed using the natural logarithm of the “odds” of the target variable, rather than the probability. Moreover, the predictors do not have to be normally distributed or have equal variance in each group.



In the logistic regression the constant (b0) moves the curve left and right and the slope (b1) defines the steepness of the curve. By simple transformation, the logistic regression equation can be written in terms of an odds ratio.

$$\frac{p}{1-p} = \exp(b_0 + b_1 x)$$

Finally, taking the natural log of both sides, we can write the equation in terms of log-odds (logit) which is a linear function of the predictors. The coefficient (b_1) is the amount the logit (log-odds) changes with a one unit change in x.

$$ln\left(\frac{p}{1-p}\right) = b_0 + b_1 x$$

As mentioned before, logistic regression can handle any number of numerical and/or categorical variables.

$$p = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p)}}$$

There are several analogies between linear regression and logistic regression. Just as ordinary least square regression is the method used to estimate coefficients for the best fit line in linear regression, logistic regression uses maximum likelihood estimation (MLE) to obtain the model coefficients that relate predictors to the target. After this initial function is estimated, the process is repeated until LL (Log Likelihood) does not change significantly.

$$\boldsymbol{\beta^1 = \beta^0 + [X^T W X]^{-1}.X^T(y-\mu)}$$

$\boldsymbol{\beta}$ *is a vector of the logistic regression coefficients.*

$\boldsymbol{W}$ *is a square matrix of order N with elements* $n_i\pi_i(1-\pi_i)$ *on the diagonal and zeros everywhere else.*

$\boldsymbol{\mu}$ *is a vector of length N with elements* $\mu_i = n_i\pi_i$.

A pseudo R^2 value is also available to indicate the adequacy of the regression model. Likelihood ratio test is a test of the significance of the difference between the likelihood ratio for the baseline model minus the likelihood ratio for a reduced model. This difference is called "model chi-square". Wald test is used to test the statistical significance of each coefficient (*b*) in the model (i.e., predictors contribution).

Pseudo R2

There are several measures intended to mimic the R2 analysis to evaluate the goodness-of-fit of logistic models, but they cannot be interpreted as one would interpret an R2 and different pseudo R2 can arrive at very different values. Here we discuss three pseudo R2measures.

Pseudo R^2	**Equation**	**Description**
Efron's	$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - p_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$	***'p'*** **is the logistic model predicted probability. The model residuals are squared, summed, and divided by the total variability in the dependent variable.**
McFadden's	$R^2 = 1 - \frac{LL_{full\ model}}{LL_{intercept}}$	**The ratio of the log-likelihoods suggests the level of improvement over the intercept model offered by the full model.**
Count	$R^2 = \frac{\#\ Corrects}{Total\ Count}$	**The number of records correctly predicted, given a cutoff point of .5 divided by the total count of cases. This is equal to the accuracy .**

Likelihood Ratio Test

The likelihood ratio test provides the means for comparing the likelihood of the data under one model (e.g., full model) against the likelihood of the data under another, more restricted model (e.g., intercept model).

$$LL = \sum_{i=1}^{n} y_i ln(p_i) + (1 - y_i) ln(1 - p_i)$$

where 'p' is the logistic model predicted probability. The next step is to calculate the difference between these two log-likelihoods.

$$2(LL_1 - LL_2)$$

The difference between two likelihoods is multiplied by a factor of 2 in order to be assessed for statistical significance using standard significance levels (Chi^2 test). The degrees of freedom for the test will equal the difference in the number of parameters being estimated under the models (e.g., full and intercept).

Disadvantages Of Existing System

- In the existing work, the system did not implement an exact sentiment analysis for large data sets.
- This system is less performance due to lack Data Classification and Data Fragmentation technique.

CHAPTER 4

PROPOSED SYSTEM

4.1 OVERVIEW

A recommender framework is a customary system that proposes an item to the user, dependent on their advantage and necessity. These frameworks employ the customers' surveys to break down their sentiment and suggest a recommendation for their exact need. In the drug recommender system, medicine is offered on a specific condition dependent on patient reviews using sentiment analysis and feature engineering. Sentiment analysis is a progression of strategies, methods, and tools for distinguishing and extracting emotional data, such as opinion and attitudes. On the other hand, featuring engineering is the process of making more features from the existing ones; it improves the performance of models.

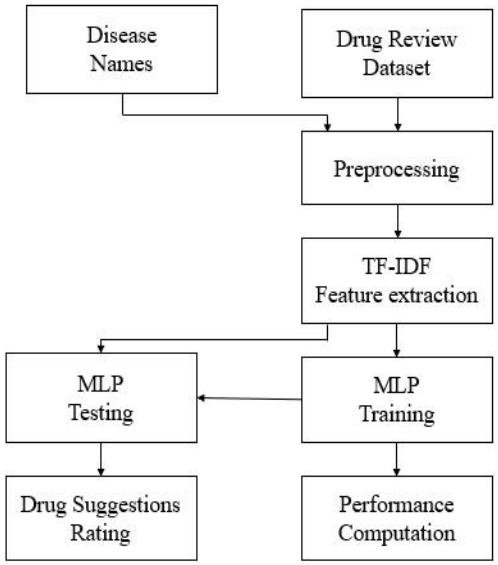


Fig. 1: Block Diagram of Proposed System

The project code appears to be a Python script utilizing the Tkinter library to create a graphical user interface (GUI) for an integrated Machine Learning (ML) with Natural Language Processing (NLP) framework aimed at a DR-SED System. Here's an overview of the project:

GUI Interface: The GUI interface is created using Tkinter, allowing users to interact with the system conveniently.

Dataset Handling:

- Users can upload a dataset containing drug reviews.
- The system preprocesses the dataset by cleaning and transforming the text data.

Feature Extraction:

- TF-IDF features are extracted from the preprocessed text data using the TfidfVectorizer from scikit-learn.

Model Training:

- Several machine learning models are trained on the dataset, including Logistic Regression, Linear SVC, Ridge Classifier, Multinomial Naive Bayes, SGD Classifier, and Multilayer Perceptron Classifier.Performance metrics such as accuracy, precision, recall, and F1-score are computed for each model and displayed to the user.

Model Persistence:

- Trained models are saved using the pickle library to avoid retraining every time the application runs.

DR-SED:

- Users can input new drug review data.
- The system recommends drugs based on the input data using the trained ML models.
- The recommendation is based on the highest similarity score between the input review and the preprocessed reviews in the dataset.

Visualization:

Graphical representations such as bar graphs are used to visualize the distribution of ratings and other performance metrics.

Custom Components:

Custom components such as TkinterCustomButton are utilized for enhanced GUI aesthetics.

Libraries Used:

The code utilizes various libraries such as matplotlib, pandas, numpy, nltk, and scikit-learn for data manipulation, visualization, NLP processing, and ML modeling.

4.2 Preprocessing

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data preprocessing task.

Need of Data Preprocessing: A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

Importing Libraries: To perform data preprocessing using Python, we need to import some predefined Python libraries. These libraries are used to perform some specific jobs. There are three specific libraries that we will use for data preprocessing, which are:

Numpy: Numpy Python library is used for including any type of mathematical operation in the code. It is the fundamental package for scientific calculation in Python. It also supports to add large, multidimensional arrays and matrices. So, in Python, we can import it as:

```
import numpy as nm
```

Here we have used nm, which is a short name for Numpy, and it will be used in the whole program.

Matplotlib: The second library is matplotlib, which is a Python 2D plotting library, and with this library, we need to import a sub-library pyplot. This library is used to plot any type of charts in Python for the code. It will be imported as below:

```
import matplotlib.pyplot as mpt
```

Here we have used mpt as a short name for this library.

Pandas: The last library is the Pandas library, which is one of the most famous Python libraries and used for importing and managing the datasets. It is an open-source data manipulation and analysis library. Here, we have used pd as a short name for this library. Consider the below image:

```
1 # importing libraries
2 import numpy as nm
3 import matplotlib.pyplot as mtp
4 import pandas as pd
5
```

Handling Missing data: The next step of data preprocessing is to handle missing data in the datasets. If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset. There are mainly two ways to handle missing data, which are:

- By deleting the particular row: The first way is used to commonly deal with null values. In this way, we just delete the specific row or column which consists of null values. But this way is not so efficient and removing data may lead to loss of information which will not give the accurate output.
- By calculating the mean: In this way, we will calculate the mean of that column or row which contains any missing value and will put it on the place of missing value. This strategy is useful for the features which have numeric data such as age, salary, year, etc.

Encoding Categorical data: Categorical data is data which has some categories such as, in our dataset; there are two categorical variables, Country, and Purchased. Since machine learning model completely works on mathematics and numbers, but if our dataset would have a categorical variable, then it may create trouble while building the model. So, it is necessary to encode these categorical variables into numbers.

Feature Scaling: Feature scaling is the final step of data preprocessing in machine learning. It is a technique to standardize the independent variables of the dataset in a specific range. In feature scaling, we put our variables in the same range and in the same scale so that no variable dominates the other variable. A machine learning model is based on Euclidean distance, and if we do not scale the variable, then it will cause some issue in our machine learning model. Euclidean distance is given as:

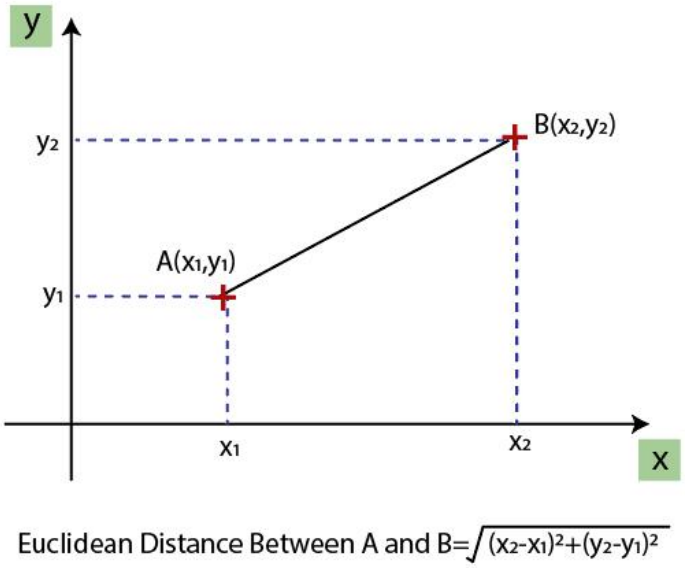
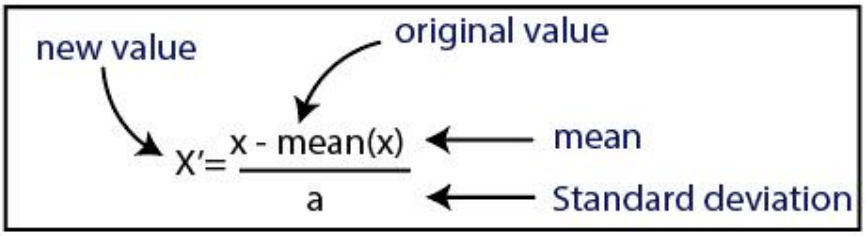


Figure 4.3. Feature scaling

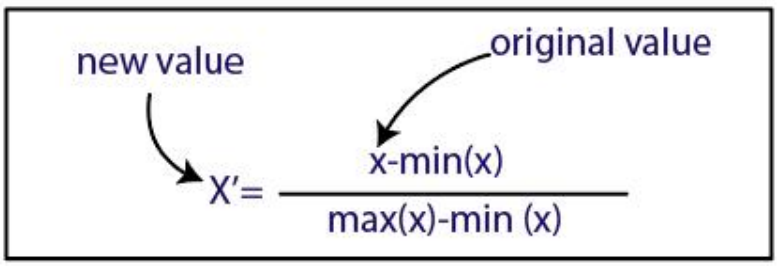
If we compute any two values from age and salary, then salary values will dominate the age values, and it will produce an incorrect result. So to remove this issue, we need to perform feature scaling for machine learning.

There are two ways to perform feature scaling in machine learning:

Standardization



Normalization



4.3 Splitting the Dataset

In machine learning data preprocessing, we divide our dataset into a training set and test set. This is one of the crucial steps of data preprocessing as by doing this, we can enhance the performance

of our machine learning model. Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models. If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:

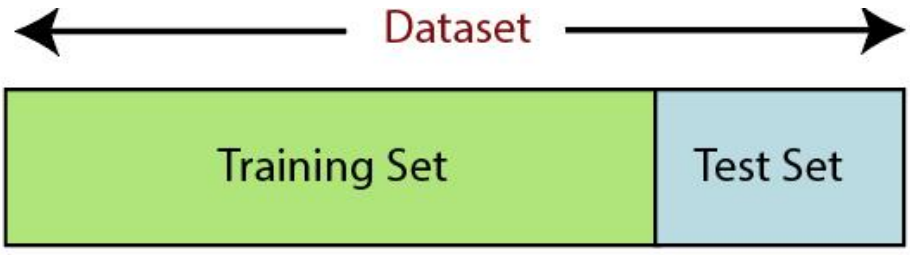


Figure 4.4. Splitting the dataset

Training Set: A subset of dataset to train the machine learning model, and we already know the output.

Test set: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

For splitting the dataset, we will use the below lines of code:

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2, random_state=0)
```

Explanation

- In the above code, the first line is used for splitting arrays of the dataset into random train and test subsets.
- In the second line, we have used four variables for our output that are
  - x_train: features for the training data
  - x_test: features for testing data
  - y_train: Dependent variables for training data
  - y_test: Independent variable for testing data
- In train_test_split() function, we have passed four parameters in which first two are for arrays of data, and test_size is for specifying the size of the test set. The test_size maybe .5, .3, or .2, which tells the dividing ratio of training and testing sets.

- The last parameter random_state is used to set a seed for a random generator so that you always get the same result, and the most used value for this is 42.

4.4 TF-IDF Feature extraction

TF-IDF which stands for Term Frequency – Inverse Document Frequency. It is one of the most important techniques used for information retrieval to represent how important a specific word or phrase is to a given document. Let's take an example, we have a string or Bag of Words (BOW) and we have to extract information from it, then we can use this approach.

The tf-idf value increases in proportion to the number of times a word appears in the document but is often offset by the frequency of the word in the corpus, which helps to adjust with respect to the fact that some words appear more frequently in general. TF-IDF use two statistical methods, first is Term Frequency and the other is Inverse Document Frequency. Term frequency refers to the total number of times a given term t appears in the document doc against (per) the total number of all words in the document and The inverse document frequency measure of how much information the word provides. It measures the weight of a given word in the entire document. IDF show how common or rare a given word is across all documents. TF-IDF can be computed as tf * idf

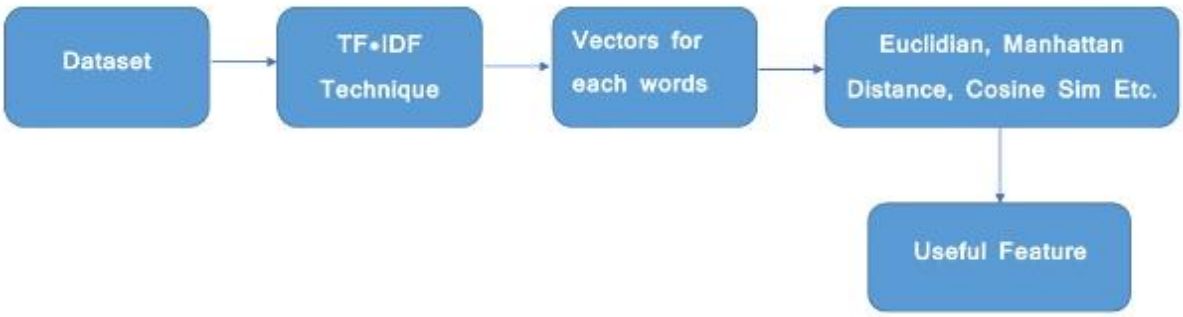


Figure 4.5: TF-IDF block diagram

TF-IDF do not convert directly raw data into useful features. Firstly, it converts raw strings or dataset into vectors and each word has its own vector. Then we'll use a particular technique for retrieving the feature like Cosine Similarity which works on vectors, etc.

Terminology:

t — term (word)

d — document (set of words)

N — count of corpus

corpus — the total document set

Term Frequency (TF): Suppose we have a set of English text documents and wish to rank which document is most relevant to the query, "Data Science is awesome!" A simple way to start out is by eliminating documents that do not contain all three words "Data" is", "Science", and "awesome", but this still leaves many documents. To further distinguish them, we might count the number of times each term occurs in each document; the number of times a term occurs in a document is called its term frequency. The weight of a term that occurs in a document is simply proportional to the term frequency.

$$tf(t,d) = count\ of\ t\ in\ d\ /\ number\ of\ words\ in\ d$$

Document Frequency: This measures the importance of document in whole set of corpus, this is very similar to TF. The only difference is that TF is frequency counter for a term t in document d, whereas DF is the count of occurrences of term t in the document set N. In other words, DF is the number of documents in which the word is present. We consider one occurrence if the term consists in the document at least once, we do not need to know the number of times the term is present.

$$df(t) = occurrence\ of\ t\ in\ documents$$

Inverse Document Frequency (IDF): While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus, we need to weigh down the frequent terms while scale up the rare ones, by computing IDF, an inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely. **The** IDF is the inverse of the document frequency which measures the informativeness of term t. When we calculate IDF, it will be very low for the most occurring words such as stop words (because stop words such as "is" is present in almost all of the documents, and N/df will give a very low value to that word). This finally gives what we want, a relative weightage.

$$idf(t) = N/df$$

Now there are few other problems with the IDF, in case of a large corpus,say 100,000,000 , the IDF value explodes , to avoid the effect we take the log of idf . During the query time, when a word which is not in vocab occurs, the df will be 0. As we cannot divide by 0, we smoothen the value by adding 1 to the denominator.

$$idf(t) = log(N/(df + 1))$$

The TF-IDF now is at the right measure to evaluate how important a word is to a document in a collection or corpus. Here are many different variations of TF-IDF but for now let us concentrate on this basic version.

$$tf - idf(t, d) = tf(t, d) * log(N/(df + 1))$$

Implementing TF-IDF: To make TF-IDF from scratch in python, let's imagine those two sentences from different document:

first_sentence : "Data Science is the sexiest job of the 21st century".

second_sentence : "machine learning is the key for data science".

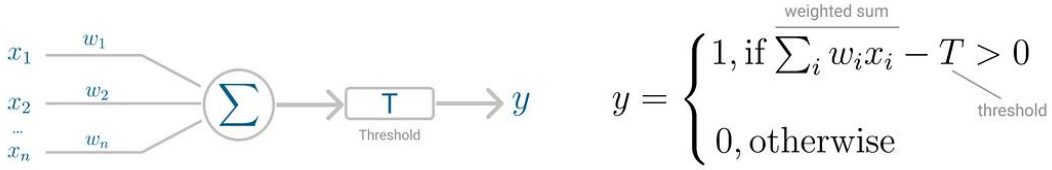
First step we have to create the TF function to calculate total word frequency for all documents.

4.5 Multilayer perceptron (MLP)

4.5.1 Perceptron

Although today the Perceptron is widely recognized as an algorithm, it was initially intended as an image recognition machine. It gets its name from performing the human-like function of perception, seeing, and recognizing images.

In particular, interest has been centered on the idea of a machine which would be capable of conceptualizing inputs impinging directly from the physical environment of light, sound, temperature, etc. — the "phenomenal world" with which we are all familiar — rather than requiring the intervention of a human agent to digest and code the necessary information. Rosenblatt's perceptron machine relied on a basic unit of computation, the neuron. Just like in previous models, each neuron has a cell that receives a series of pairs of inputs and weights. The major difference in Rosenblatt's model is that inputs are combined in a weighted sum and, if the weighted sum exceeds a predefined threshold, the neuron fires and produces an output.



Perceptron neuron model (left) and threshold logic (right).

Threshold T represents the activation function. If the weighted sum of the inputs is greater than zero the neuron outputs the value 1, otherwise the output value is zero.

Perceptron for Binary Classification

With this discrete output, controlled by the activation function, the perceptron can be used as a binary classification model, defining a linear decision boundary.

It finds the separating hyperplane that minimizes the distance between misclassified points and the decision boundary. The perceptron loss function is defined as below:

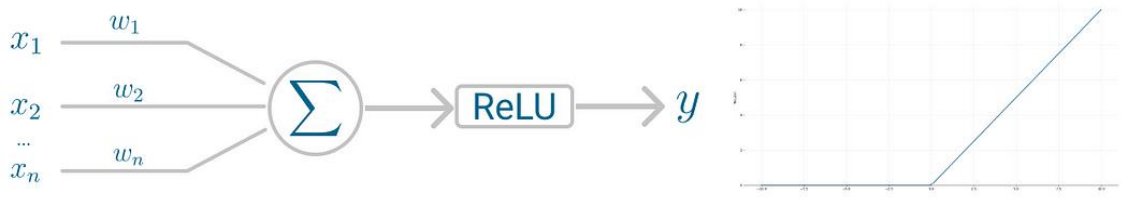
$$\underbrace{D(w,c)}_{\text{distance}} = -\sum_{i \in \mathrm{M}} \overbrace{y_i}^{\text{output}} (x_i w_i + c)$$

misclassified observations

To minimize this distance, perceptron uses stochastic gradient descent (SGD) as the optimization function. If the data is linearly separable, it is guaranteed that SGD will converge in a finite number of steps. The last piece that Perceptron needs is the activation function, the function that determines if the neuron will fire or not. Initial Perceptron models used sigmoid function, and just by looking at its shape, it makes a lot of sense! The sigmoid function maps any real input to a value that is either 0 or 1 and encodes a non-linear function. The neuron can receive negative numbers as input, and it will still be able to produce an output that is either 0 or 1.

But, if you look at Deep Learning papers and algorithms from the last decade, you'll see the most of them use the Rectified Linear Unit (ReLU) as the neuron's activation function. The reason why ReLU became more adopted is that it allows better optimization using SGD, more efficient computation and is scale-invariant, meaning, its characteristics are not affected by the scale of the input.

The neuron receives inputs and picks an initial set of weights random. These are combined in weighted sum and then ReLU, the activation function, determines the value of the output.

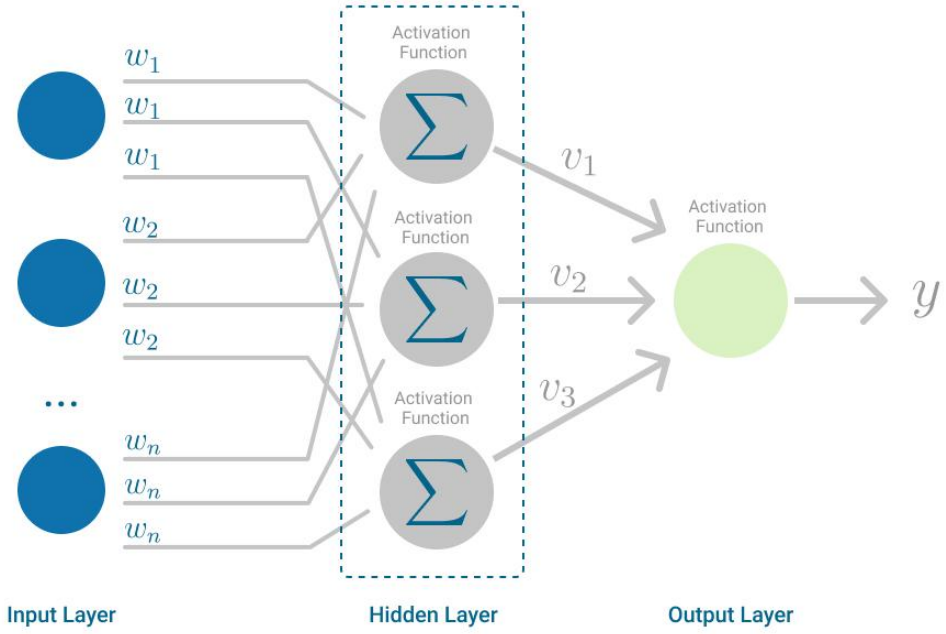


Perceptron neuron model (left) and activation function (right).

Perceptron uses SGD to find, or you might say learn, the set of weight that minimizes the distance between the misclassified points and the decision boundary. Once SGD converges, the dataset is separated into two regions by a linear hyperplane. Although it was said the Perceptron could represent any circuit and logic, the biggest criticism was that it couldn't represent the XOR gate, exclusive OR, where the gate only returns 1 if the inputs are different. This was proved almost a decade later and highlights the fact that Perceptron, with only one neuron, can't be applied to non-linear data.

4.5.2 MLP

The MLP was developed to tackle this limitation. It is a neural network where the mapping between inputs and output is non-linear. A MLP has input and output layers, and one or more hidden layers with many neurons stacked together. And while in the Perceptron the neuron must have an activation function that imposes a threshold, like ReLU or sigmoid, neurons in a MLP can use any arbitrary activation function.



Architecture of MLP.

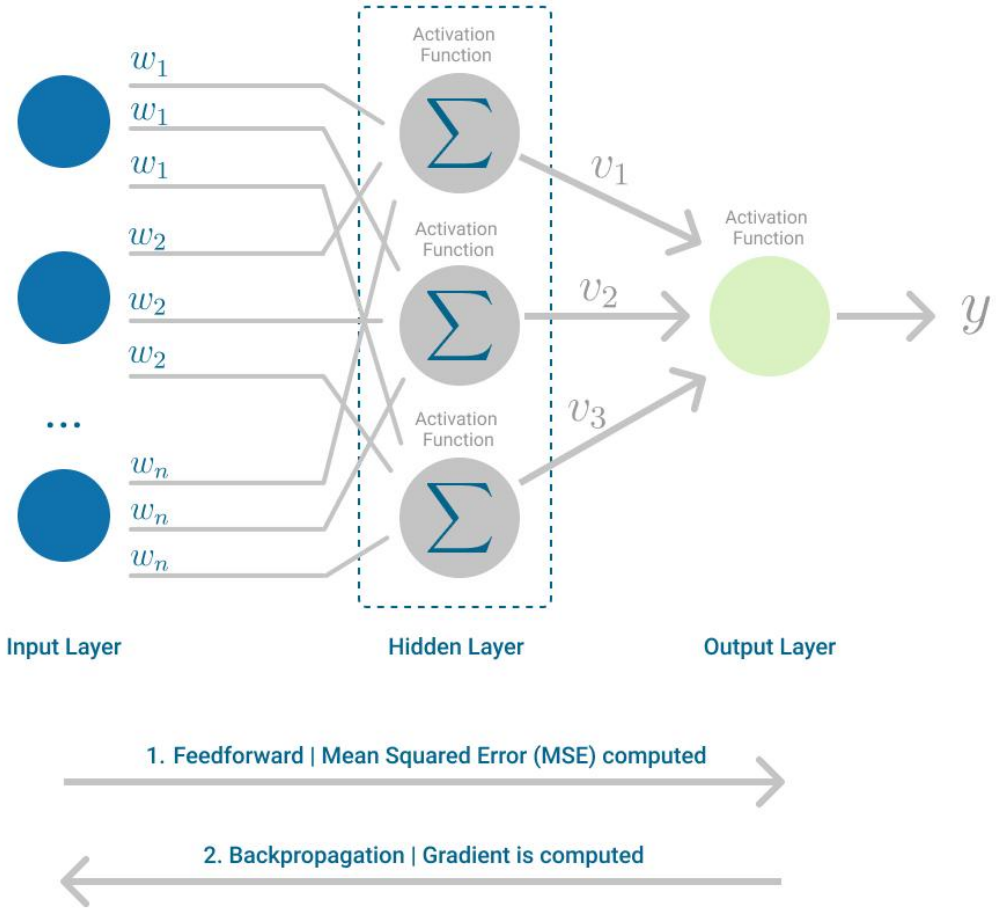
MLP falls under the category of feedforward algorithms, because inputs are combined with the initial weights in a weighted sum and subjected to the activation function, just like in the Perceptron. But the difference is that each linear combination is propagated to the next layer. Each layer is feeding the next one with the result of their computation, their internal representation of the data. This goes all the way through the hidden layers to the output layer.

If the algorithm only computed the weighted sums in each neuron, propagated results to the output layer, and stopped there, it wouldn't be able to learn the weights that minimize the cost function. If the algorithm only computed one iteration, there would be no actual learning. This is where Backpropagation comes into play.

Backpropagation

Backpropagation is the learning mechanism that allows the MLP to iteratively adjust the weights in the network, with the goal of minimizing the cost function. There is one hard requirement for backpropagation to work properly.

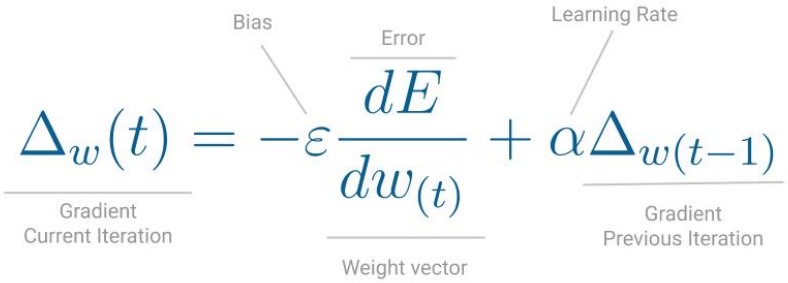
The function that combines inputs and weights in a neuron, for instance the weighted sum, and the threshold function, for instance ReLU, must be differentiable. These functions must have a bounded derivative because Gradient Descent is typically the optimization function used in MLP.



MLP, highlighting the Feedforward and Backpropagation steps.

In each iteration, after the weighted sums are forwarded through all layers, the gradient of the Mean Squared Error is computed across all input and output pairs. Then, to propagate it back,

the weights of the first hidden layer are updated with the value of the gradient. That's how the weights are propagated back to the starting point of the neural network. One iteration of Gradient Descent is defined as follows:



This process keeps going until gradient for each input-output pair has converged, meaning the newly computed gradient hasn't changed more than a specified convergence threshold, compared to the previous iteration

CHAPTER 5

UML DAIGRAMS

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.

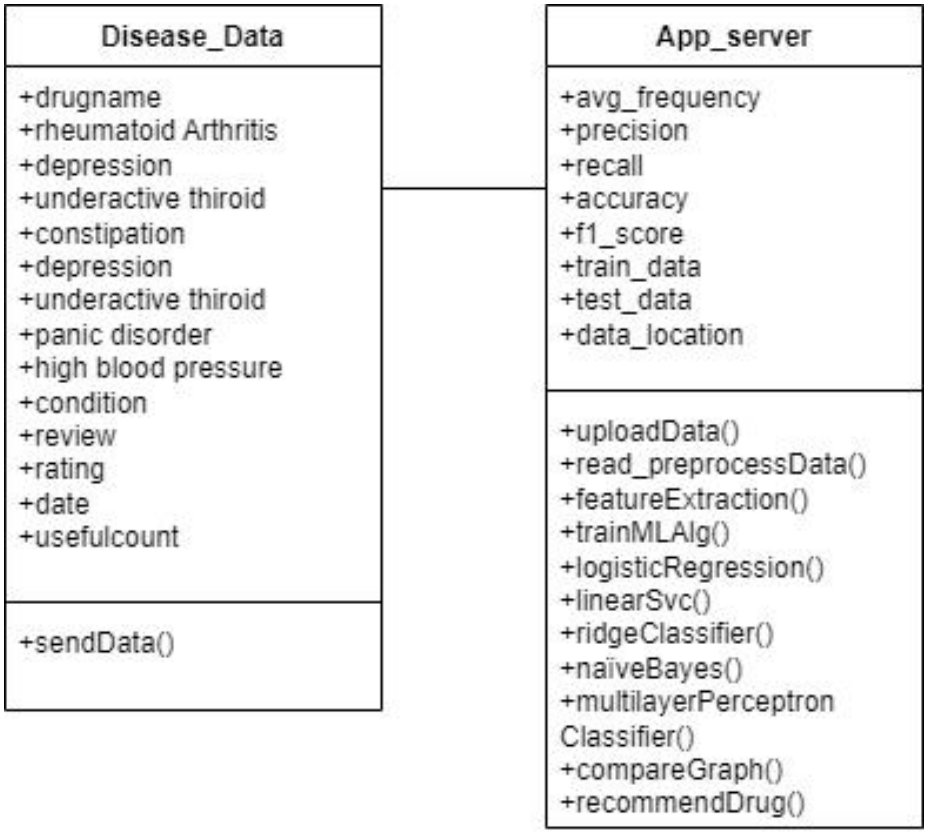
The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful 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:

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

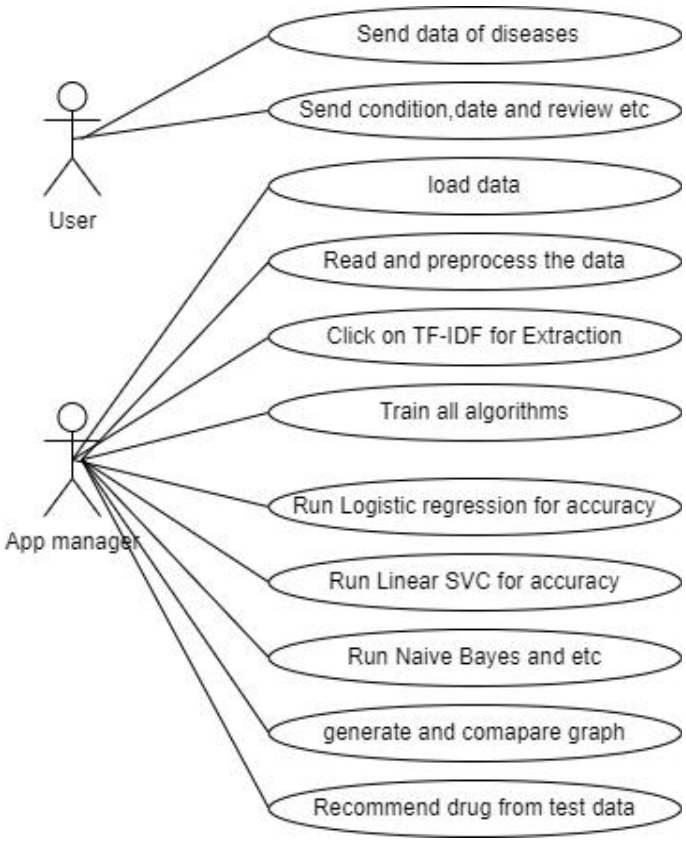
Class diagram

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.



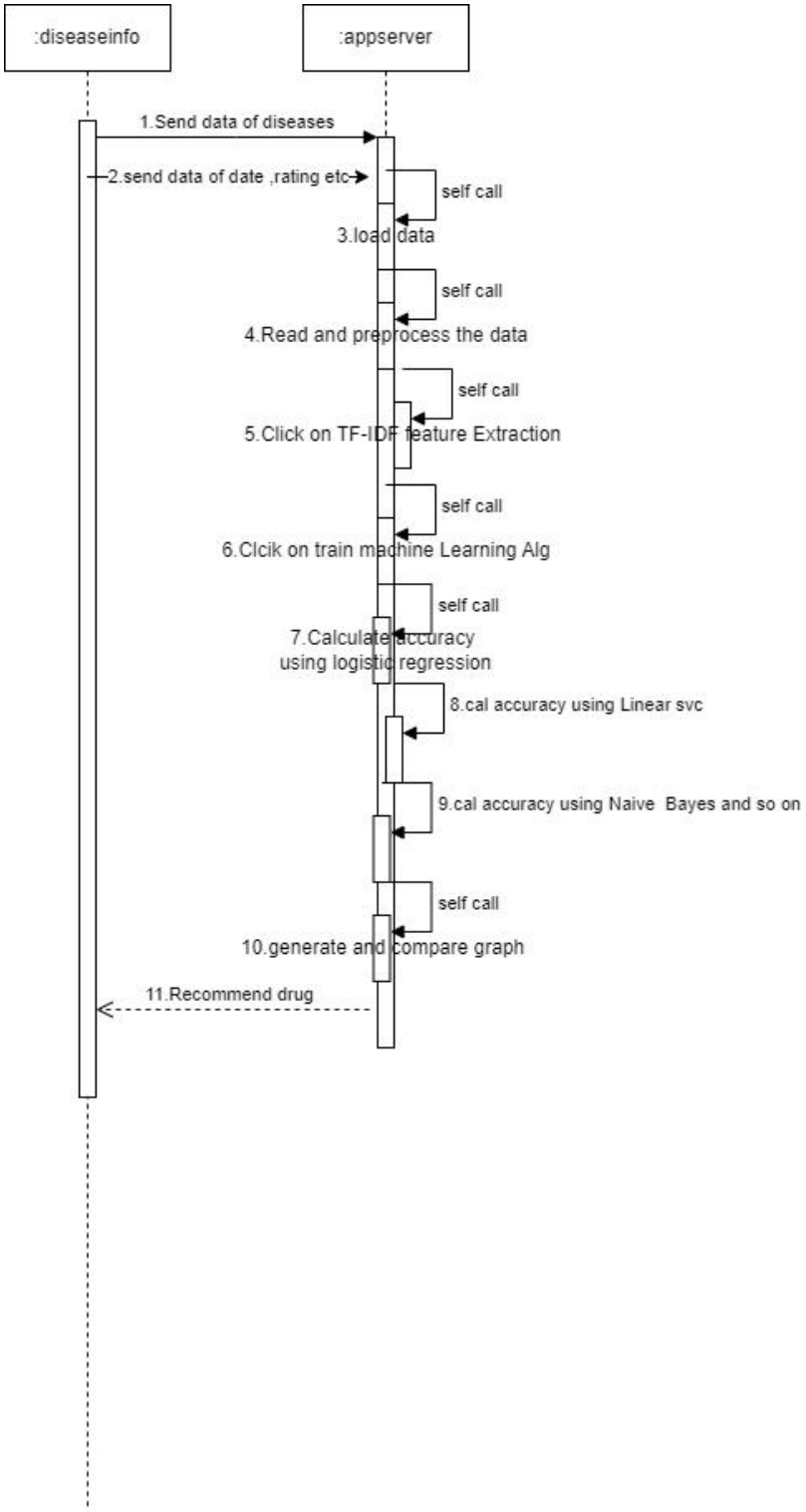
Use case diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



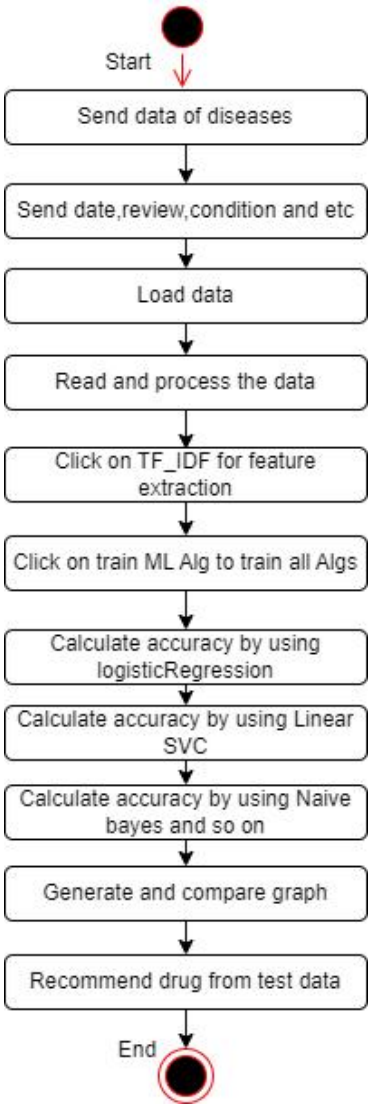
Sequence diagram

A sequence diagram in Unified Modeling Language (UML) 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, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

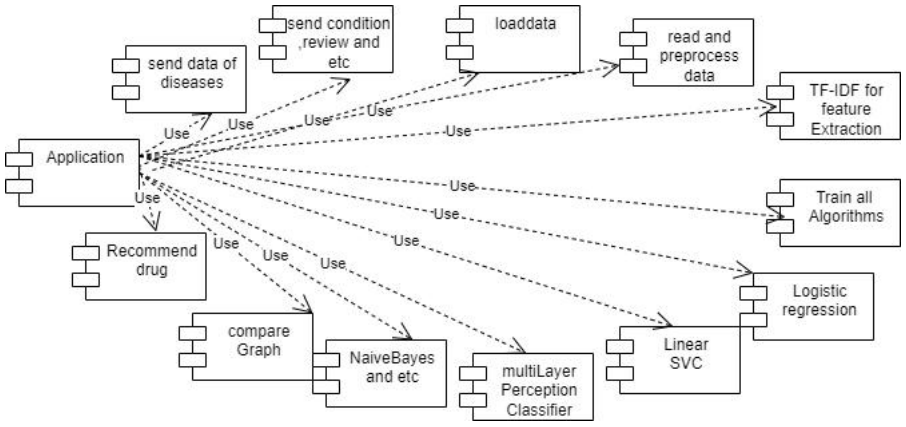


Activity diagram

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



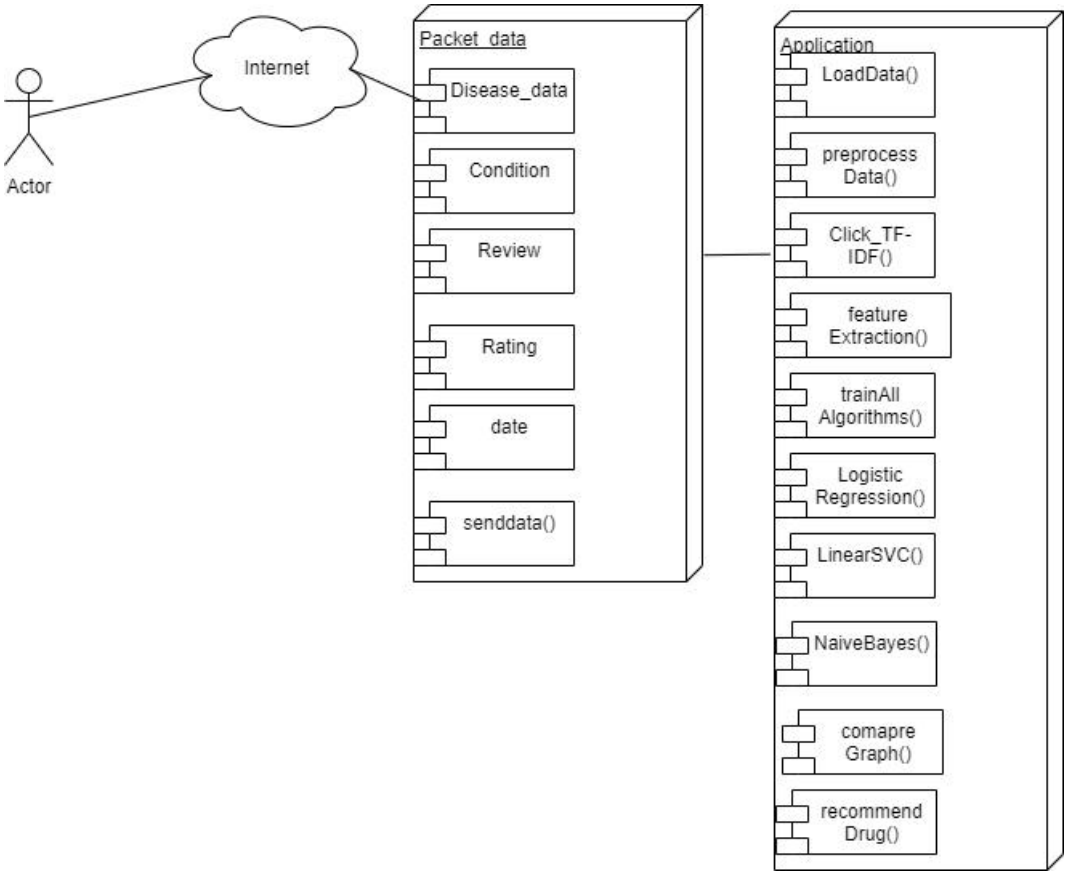
Component diagram: Component diagram describes the organization and wiring of the physical components in a system.



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.



CHAPTER 6

MACHINE LEARNING

What is Machine Learning

Before we look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Categories of Machine Leaning

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data,

while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate, and solve complex problems. On the other side, AI is still in its initial stage and have not surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, “to make decisions, based on data, with efficiency and scale”.

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programming logic, in the problems that cannot be programmed inherently. The fact is that we can’t do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

Challenges in Machines Learning

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

1. Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.
2. Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.
3. Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.
4. No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.
5. Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.

6. Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.
7. Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

Applications of Machines Learning

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

How to Start Learning Machine Learning?

Arthur Samuel coined the term “Machine Learning” in 1959 and defined it as a “Field of study that gives computers the capability to learn without being explicitly programmed”.

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a 344% growth and an average base salary of $146,085 per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So, this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let’s get started!!!

How to start learning ML?

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don't know these, never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!!
Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

So if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as Fork Python available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

(a) Terminologies of Machine Learning

- Model – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- Feature – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
- Target (Label) – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- Training – The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- Prediction – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Machine Learning

- Supervised Learning – This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- Unsupervised Learning – This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- Semi-supervised Learning – This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- Reinforcement Learning – This involves learning optimal actions through trial and error.

So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

Advantages of Machine learning

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

Disadvantages of Machine Learning

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

CHAPTER 7

SOFTWARE ENVIRONMENT

What is Python?

Below are some facts about Python.

- Python is currently the most widely used multi-purpose, high-level programming language.
- Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
- Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
- Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python

Let’s see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

8. Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

9. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

Advantages of Python Over Other Languages

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is dynamically typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

History of Python

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners1, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

Python Development Steps

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system.

Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

Print is now a function.

- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e., int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
- Text Vs. Data Instead of Unicode Vs. 8-bit

Purpose

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python

excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Modules Used in Project

TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas

is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and I Python shells, the Jupiter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about

how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here.

The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

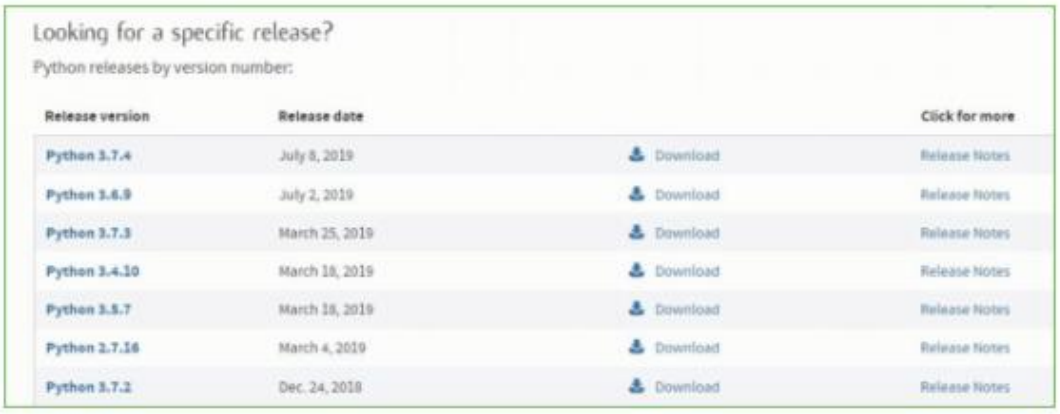
Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4



Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e52d2db4aef7b9ab010f0f9be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae66097051c2eca45ee3604803	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583daff1a442cbalcee08e6	34898416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a457738f5e4a936b241f	28082845	SIG
Windows help file	Windows		d63999573a2c56b2ac56cade6b4f7cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b00c3cf6d9ec0b9abe83184a4072fa2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76de6db3043a583e563400	26880368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c6088bd73ae8e53a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3bd1f8b41879fda9413357413 9d8	6741628	SIG
Windows x86 executable installer	Windows		33cc80294 2a5444da3d64514769 4789	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa5d317df82c3093 3ea371d87c	1324608	SIG

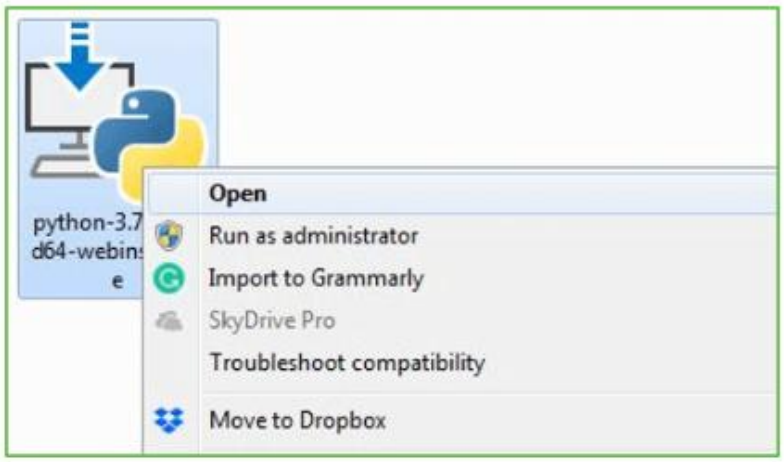
- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

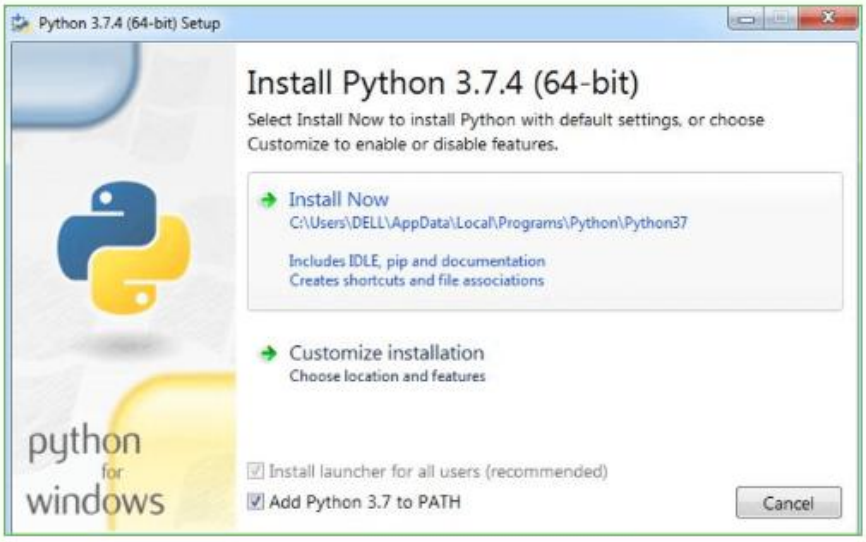
Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.



Step 3: Click on Install NOW After the installation is successful. Click on Close.

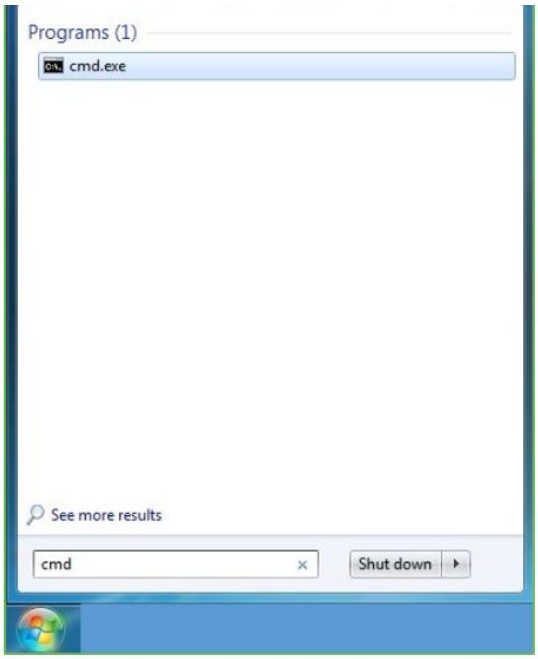
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

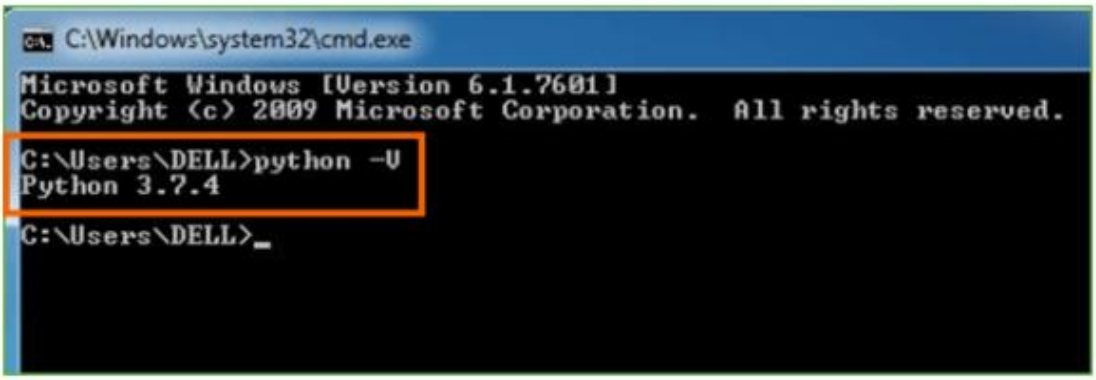
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.



Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

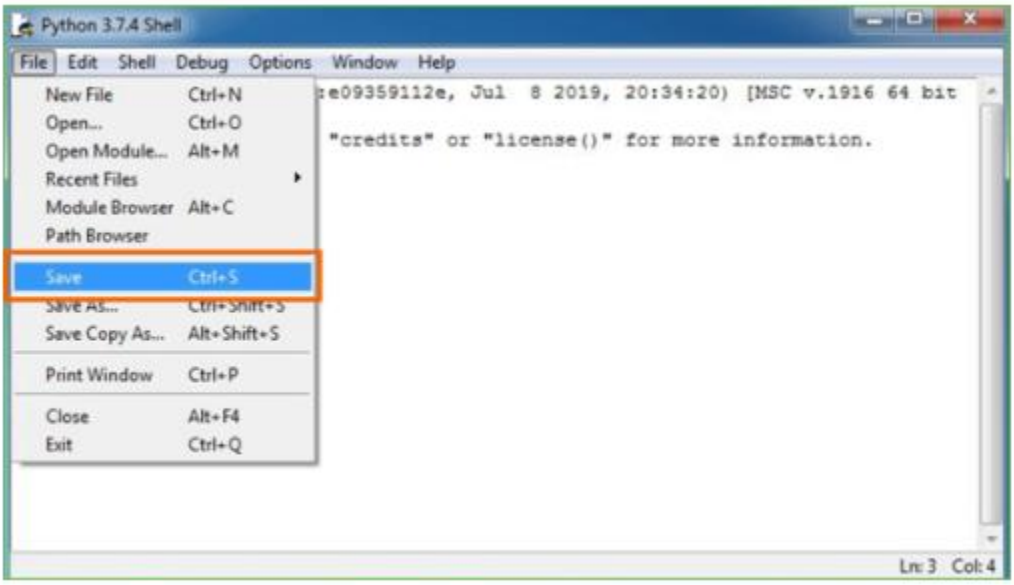
Check how the Python IDLE works

Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.

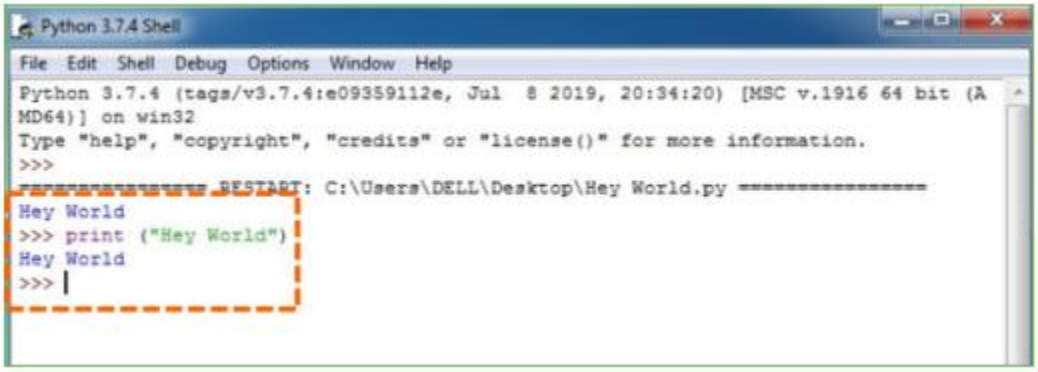
Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter print ("Hey World") and Press Enter.



You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won't work.

CHAPTER 8

SYSTEM REQUIREMENTS SPECIFICATIONS

Software Requirements

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

- Python IDLE 3.7 version (or)
- Anaconda 3.7 (or)
- Jupiter (or)
- Google colab

Hardware Requirements

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

Operating system	:	Windows, Linux
Processor	:	minimum intel i3
Ram	:	minimum 4 GB
Hard disk	:	minimum 250GB

CHAPTER 9

FUNCTIONAL REQUIREMENTS

Output Design

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization
- Internal Outputs whose destination is within organization and they are the
- User's main interface with the computer.
- Operational outputs whose use is purely within the computer department.
- Interface outputs, which involve the user in communicating directly.

Output Definition

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

Input Design

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

Input Stages

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

Input Types

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to the system?
- Interactive, which are inputs entered during a dialogue.

Input Media

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

Error Avoidance

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

Error Detection

Even though every effort is made to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

Data Validation

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

User Interface Design

It is essential to consult the system users and discuss their needs while designing the user interface:

User Interface Systems Can Be Broadly Clasified As:

- User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
- Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

User Initiated Intergfaces

User initiated interfaces fall into two approximate classes:

- Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
- Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

Computer-Initiated Interfaces

The following computer – initiated interfaces were used:

- The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
- Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

Error Message Design

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

Performance Requirements

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system
- The existing system is completely dependent on the user to perform all the duties.

CHAPTER 10

SOURCE CODE

```
from tkinter import messagebox

from tkinter import *

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

import matplotlib.pyplot as plt

from tkinter.filedialog import askopenfilename

from CustomButton import TkinterCustomButton

from sklearn.model_selection import train_test_split

from sklearn.metrics import precision_score

from sklearn.metrics import recall_score

from sklearn.metrics import f1_score

from sklearn.metrics import accuracy_score

import pandas as pd

import numpy as np

import pickle

import os


from string import punctuation

from nltk.corpus import stopwords

import nltk

from nltk.stem import WordNetLemmatizer

from sklearn.feature_extraction.text import TfidfVectorizer
```

```
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import RidgeClassifier
from numpy import dot
from numpy.linalg import norm
from serpapi import GoogleSearch
import textwrap



main = Tk()
main.title("Enhancing Drug Side Effect Prediction with XAI for Medical Health Applications")
main.geometry("1300x1200")

global filename
global dataset
global X, Y
global X_train, X_test, y_train, y_test
accuracy = []
precision = []
recall = []
fscore = []
```

```
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

global drug_name, condition, review, rating
global tfidf_vectorizer
global classifier

def cleanPost(doc):
    tokens = doc.split()
    table = str.maketrans('', '', punctuation)
    tokens = [w.translate(table) for w in tokens]
    tokens = [word for word in tokens if word.isalpha()]
    tokens = [w for w in tokens if not w in stop_words]
    tokens = [word for word in tokens if len(word) > 1]
    tokens = [lemmatizer.lemmatize(token) for token in tokens]
    tokens = ' '.join(tokens)
    return tokens

def uploadDataset():
    global filename
    global dataset
    text.delete('1.0', END)
    filename = askopenfilename(initialdir = "Dataset")
    tf1.insert(END,str(filename))
    text.insert(END,"Dataset Loaded\n\n")
```

```
    dataset = pd.read_csv(filename,sep="\t",nrows=5000)
    text.insert(END,str(dataset.head()))
    label = dataset.groupby('rating').size()
    label.plot(kind="bar")
    plt.title("Ratings Graph")
    plt.show()


def preprocessDataset():
    global X, Y
    global X_train, X_test, y_train, y_test
    global drug_name, condition, review, rating
    global dataset
    text.delete('1.0', END)
    if os.path.exists('model/data.npy'):
        data = np.load("model/data.npy")
        drug_name = data[0]
        condition = data[1]
        review = data[2]
        rating = data[3]
    else:
        for i in range(len(dataset)):
            dname = dataset.get_value(i,"drugName")
            cond = dataset.get_value(i,"condition")
            reviewText = dataset.get_value(i,"review")
```

```
            ratings = dataset.get_value(i,"rating")
            reviewText = str(reviewText)
            reviewText = reviewText.strip().lower()
            reviewText = cleanPost(reviewText)
            drug_name.append(dname)
            condition.append(cond)
            review.append(reviewText)
            rating.append(ratings-1)
            print(i)
        data = [drug_name,condition,review,rating]
        data = np.asarray(data)
        np.save("model/data",data)
    text.insert(END,"Reviews after cleaning and preprocessing\n\n")
    text.insert(END,str(review))
    label = dataset.groupby('drugName').size().head(20)
    label.plot(kind="bar")
    plt.title("Top 20 Drug Name Graph")
    plt.show()

def TFIDFExtraction():
    global drug_name, condition, review, rating
    global tfidf_vectorizer
    text.delete('1.0', END)
    global X, Y
    global X_train, X_test, y_train, y_test
```

```
    tfidf_vectorizer = TfidfVectorizer(stop_words=stop_words, use_idf=True, smooth_idf=False,
norm=None, decode_error='replace', max_features=700)
    tfidf = tfidf_vectorizer.fit_transform(review).toarray()
    df = pd.DataFrame(tfidf, columns=tfidf_vectorizer.get_feature_names())
    text.insert(END,str(df)+"\n\n")
    text.insert(END,str(df.values[0]))
    df = df.values
    X = df[:, 0:df.shape[1]]
    Y = rating
    indices = np.arange(X.shape[0])
    np.random.shuffle(indices)
    X = X[indices]
    Y = Y[indices]
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)



def test(cls,name):
    predict = cls.predict(X_test)
    acc = accuracy_score(y_test,predict)*100
    p = precision_score(y_test,predict,average='macro') * 100
    r = recall_score(y_test,predict,average='macro') * 100
    f = f1_score(y_test,predict,average='macro') * 100
    text.insert(END,name+" Precision  : "+str(p)+"\n")
    text.insert(END,name+" Recall     : "+str(r)+"\n")
    text.insert(END,name+" F1-Score   : "+str(f)+"\n")
    text.insert(END,name+" Accuracy   : "+str(acc)+"\n\n")
```

```
    precision.append(p)
    accuracy.append(acc)
    recall.append(r)
    fscore.append(f)

def TrainML():
    global X, Y
    global X_train, X_test, y_train, y_test
    global classifier
    text.delete('1.0', END)
    if os.path.exists('model/lr.txt'):
        with open('model/lr.txt', 'rb') as file:
            lr_cls = pickle.load(file)
        file.close()
        test(lr_cls,"Logistic Regression")
    else:
        lr_cls = LogisticRegression(max_iter=500)
        lr_cls.fit(X,Y)
        test(lr_cls,"Logistic Regression")
        with open('model/lr.txt', 'wb') as file:
            pickle.dump(lr_cls, file)
        file.close()

    if os.path.exists('model/svc.txt'):
        with open('model/svc.txt', 'rb') as file:
```

```
        svc_cls = pickle.load(file)
    file.close()
    test(svc_cls,"Linear SVC")
else:
    svc_cls = LinearSVC()
    svc_cls.fit(X,Y)
    test(svc_cls,"Linear SVC")
    with open('model/svc.txt', 'wb') as file:
        pickle.dump(svc_cls, file)
    file.close()

if os.path.exists('model/ridge.txt'):
    with open('model/ridge.txt', 'rb') as file:
        ridge_cls = pickle.load(file)
    file.close()
    test(ridge_cls,"Ridge Classifier")
else:
    ridge_cls = RidgeClassifier()
    ridge_cls.fit(X,Y)
    test(ridge_cls,"Ridge Classifier")
    with open('model/ridge.txt', 'wb') as file:
        pickle.dump(ridge_cls, file)
    file.close()

if os.path.exists('model/nb.txt'):
```

```
    with open('model/nb.txt', 'rb') as file:
        nb_cls = pickle.load(file)
    file.close()
    test(nb_cls,"Multinomial Naive Bayes")
else:
    nb_cls = MultinomialNB()
    nb_cls.fit(X,Y)
    test(nb_cls,"Multinomial Naive Bayes")
    with open('model/nb.txt', 'wb') as file:
        pickle.dump(nb_cls, file)
    file.close()

if os.path.exists('model/sgd.txt'):
    with open('model/sgd.txt', 'rb') as file:
        sgd_cls = pickle.load(file)
    file.close()
    test(sgd_cls,"SGDClassifier")
else:
    sgd_cls = MultinomialNB()
    sgd_cls.fit(X,Y)
    test(sgd_cls,"SGDClassifier")
    with open('model/sgd.txt', 'wb') as file:
        pickle.dump(sgd_cls, file)
    file.close()
```

```
    if os.path.exists('model/mlp.txt'):
        with open('model/mlp.txt', 'rb') as file:
            mlp_cls = pickle.load(file)
        file.close()
        test(mlp_cls,"Multilayer Perceptron Classifier")
        classifier = mlp_cls
    else:
        mlp_cls = MLPClassifier()
        mlp_cls.fit(X,Y)
        test(mlp_cls,"Multilayer Perceptron Classifier")
        with open('model/mlp.txt', 'wb') as file:
            pickle.dump(mlp_cls, file)
        file.close()
        classifier = mlp_cls



def graph():
    df = pd.DataFrame([['Logistic Regression','Accuracy',accuracy[0]],['Logistic
Regression','Precision',precision[0]],['Logistic Regression','Recall',recall[0]],['Logistic
Regression','FScore',fscore[0]],
                       ['Linear SVC','Accuracy',accuracy[1]],['Linear
SVC','Precision',precision[1]],['Linear SVC','Recall',recall[1]],['Linear SVC','FScore',fscore[1]],
                       ['Ridge Classifier','Accuracy',accuracy[2]],['Ridge
Classifier','Precision',precision[2]],['Ridge Classifier','Recall',recall[2]],['Ridge
Classifier','FScore',fscore[2]],
```

```
['MultinomialNB','Accuracy',accuracy[3]],['MultinomialNB','Precision',precision[3]],['Multinomia
lNB','Recall',recall[3]],['MultinomialNB','FScore',fscore[3]],

['SGDClassifier','Accuracy',accuracy[4]],['SGDClassifier','Precision',precision[4]],['SGDClassifier
','Recall',recall[4]],['SGDClassifier','FScore',fscore[4]],

                ['MLP Classifier','Accuracy',accuracy[5]],['MLP
Classifier','Precision',precision[5]],['MLP Classifier','Recall',recall[5]],['MLP
Classifier','FScore',fscore[5]],


                ],columns=['Parameters','Algorithms','Value'])
  df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar')
  plt.show()


def chat_bot(query):
  # SerpApi API key (replace with your own key)
  API_KEY = "2002589b111da86c315f882997a2e5e1f4322fb9558829945f58f8af8e553f40"


  # Check if the query is a drug name for side effects
  if query.lower().startswith("side effects of "):
    drug_name = query[15:].strip()  # Extract drug name
    search_query = f"{drug_name} side effects site:*.gov"  # Restrict to trusted .gov sites
  else:
    search_query = query  # Use the original query


  # Search parameters
  params = {
```

```
    "q": search_query,
    "hl": "en",
    "gl": "us",
    "api_key": API_KEY
}

try:
    # Initialize GoogleSearch
    search = GoogleSearch(params)
    results = search.get_dict()

    # Extract snippets from search results
    if "organic_results" in results:
        snippets = []
        for result in results["organic_results"]:
            snippet = result.get("snippet", "")
            source = result.get("source", "Unknown source")
            if snippet:
                snippets.append(f"Source: {source}\n{snippet}")

        # Combine snippets into paragraphs with drug name
        if query.lower().startswith("side effects of "):
            header = f"Side Effects for {drug_name}:\n"
        else:
            header = f"Results for '{query}':\n"
```

```
            paragraphs = header + "\n".join(textwrap.fill(snippet, width=100) for snippet in snippets)
            return paragraphs
        else:
            return f"No side effect information found for '{query}'."
    except Exception as e:
        return f"Error fetching side effects for '{query}': {str(e)}"



def recommendDrug():
    text.delete('1.0', END)
    global X
    global drug_name, condition, review, rating
    global classifier
    global tfidf_vectorizer
    filename = askopenfilename(initialdir="Dataset")
    testData = pd.read_csv(filename)
    testData = testData.values
    for i in range(len(testData)):
        review = cleanPost(testData[i,0].strip().lower())
        array = tfidf_vectorizer.transform([review]).toarray()
        predict = classifier.predict(array)[0]
        maxValue = 0
        dname = "none"
        print(str(array[0].shape)+" "+str(X.shape))
        for j in range(len(X)):
```

```
            score = dot(X[j], array[0])/(norm(X[j])*norm(array[0]))
            if score > maxValue:
                maxValue = score
                dname = drug_name[j]
        text.insert(END, f"Disease Name: {testData[i,0]}\n")
        text.insert(END, f"Recommended Drug: {dname}\n")
        text.insert(END, f"Predicted Ratings: {predict}\n")
        # Call chat_bot to get side effects for the recommended drug
        if dname != "none":
            side_effects = chat_bot(f"side effects of {dname}")
            text.insert(END, f"{side_effects}\n\n")
        else:
            text.insert(END, "No side effect information available (no drug recommended).\n\n")



font = ('times', 15, 'bold')
title = Label(main, text='Enhancing Drug Side Effect Prediction with XAI for Medical Health Applications')
title.config(bg='HotPink4', fg='yellow2')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)


font1 = ('times', 13, 'bold')
ff = ('times', 12, 'bold')
```

```
l1 = Label(main, text='Dataset Location:')
l1.config(font=font1)
l1.place(x=50,y=100)

tf1 = Entry(main,width=60)
tf1.config(font=font1)
tf1.place(x=230,y=100)
uploadButton = TkinterCustomButton(text="Upload Drug Review Dataset", width=300,
corner_radius=5, command=uploadDataset)
uploadButton.place(x=50,y=150)

preprocessButton = TkinterCustomButton(text="Read & Preprocess Dataset", width=300,
corner_radius=5, command=preprocessDataset)
preprocessButton.place(x=400,y=150)
tfidfButton = TkinterCustomButton(text="TF-IDF Features Extraction", width=300,
corner_radius=5, command=TFIDFExtraction)
tfidfButton.place(x=790,y=150)
trainMLButton = TkinterCustomButton(text="Train XAI Algorithms", width=300,
corner_radius=5, command=TrainML)
trainMLButton.place(x=50,y=200)
graphButton = TkinterCustomButton(text="Comparison Graph", width=300, corner_radius=5,
command=graph)
graphButton.place(x=400,y=200)
predictButton = TkinterCustomButton(text="Recommend Drug from Test Data", width=300,
corner_radius=5, command=recommendDrug)
predictButton.place(x=790,y=200)
font1 = ('times', 13, 'bold')
```

```
text=Text(main,height=20,width=130)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=250)

text.config(font=font1)

main.config(bg='plum2')

main.mainloop()
```

CHAPTER 11

RESULTS AND DISCUSSION

10.1 Implement Description

To implement this project, we have designed following modules

- **GUI Initialization (Tkinter)**
  - Sets up a window titled *"Enhancing Drug Side Effect Prediction with XAI…"*.
  - Adds buttons and text areas for interaction.
- **Dataset Upload & Preprocessing**
  - Loads a `.tsv` file using `pandas.read_csv()`.
  - Preprocesses drug review text using tokenization, stopword removal, punctuation stripping, and lemmatization.
  - Stores processed data as a `.npy` file to avoid repeated computation.
- **Feature Extraction**
  - Uses `TfidfVectorizer` (TF-IDF) to convert text into numerical vectors.
  - Limits vocabulary to top 700 features for efficiency.
- **Model Training**
  - Trains six classifiers:
    - Logistic Regression
    - Linear SVC
    - Ridge Classifier
    - Multinomial Naive Bayes
    - SGD Classifier
    - MLP Classifier
  - Models are persisted using `pickle`.
- **Performance Evaluation**
  - Calculates and displays:

  - Accuracy
  - Precision
  - Recall
  - F1-Score
- Plots bar graphs to compare performance of all classifiers.

- **Placeholder for Chatbot**
  - A function `chat_bot(query)` is introduced but incomplete. It seems intended to use **SerpApi** for querying drug-related information.

10.2 Dataset Description

Figure 1 represents dataset column names such as drug name, condition, review and rating and remaining rows contains dataset values and we will used above REVIEWS and RATINGS to trained machine learning models. The dataset provides patient reviews on specific drugs along with related conditions and a 10-star patient rating reflecting overall patient satisfaction. The data was obtained by crawling online pharmaceutical review sites. The intention was to study

1. sentiment analysis of drug experience over multiple facets, i.e. sentiments learned on specific aspects such as effectiveness and side effects,
2. the transferability of models among domains, i.e. conditions, and
3. the transferability of models among different data sources (see 'Drug Review Dataset (Druglib.com)').

Attribute Information

- drugName (categorical): name of drug
- condition (categorical): name of condition
- review (text): patient review
- rating (numerical): 10 star patient rating
- date (date): date of review entry
- usefulCount (numerical): number of users who found review useful

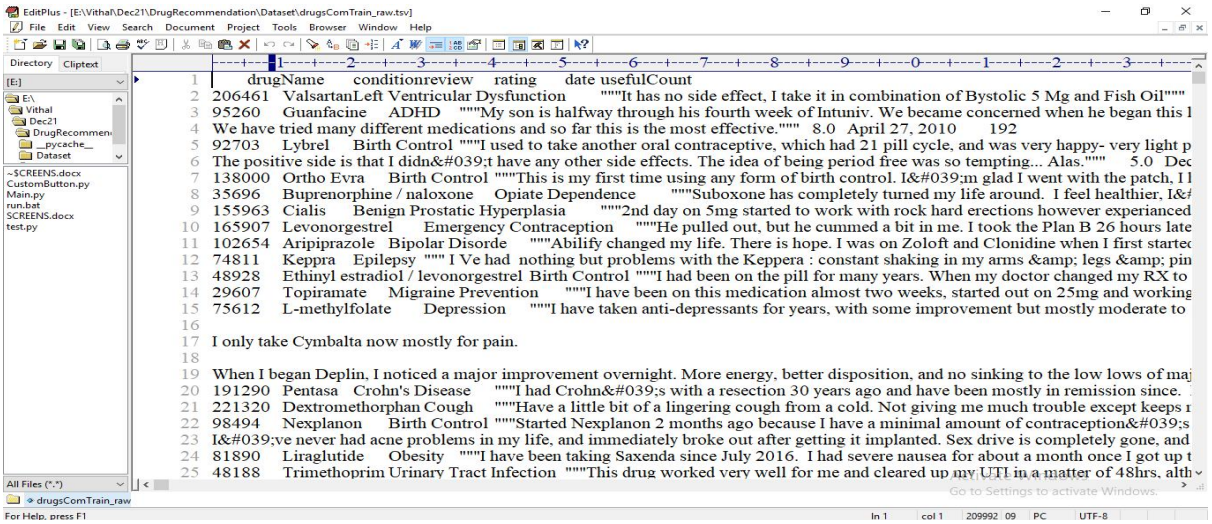


Figure 1. Displays the Sample dataset.

10.3 Results Description

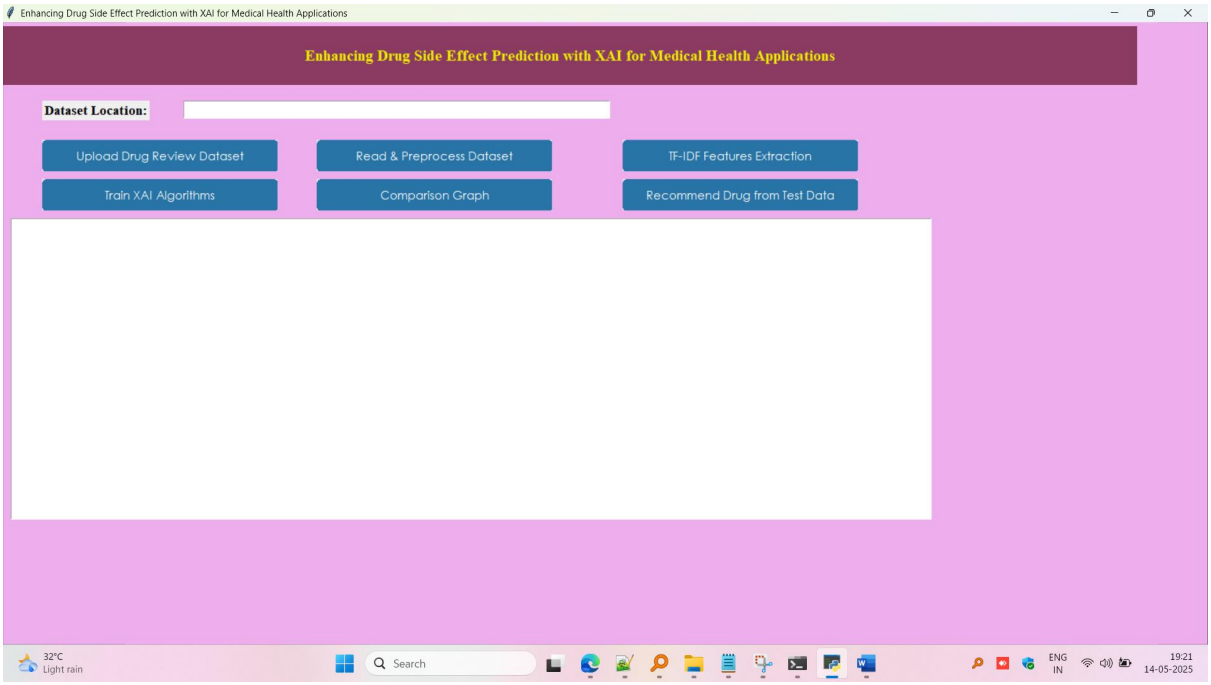


Figure 2. UI Screen of Research Work.

Figure 2 shows the UI screen of research work. Figure 3 illustrates the dataset after it has been loaded into the research environment, likely within a software tool or programming interface used for analysis. This figure confirms that the dataset, with columns like drugName, condition, review, rating, date, and usefulCount, has been successfully imported and is ready for processing. The loaded dataset mirrors the structure shown in Figure 1, ensuring all attributes are intact for further steps such as natural language processing (NLP) or machine learning (ML) model training. This figure is critical as it verifies the integrity of the data ingestion process, a prerequisite for reliable analysis.

Figure 3. Sample Dataset Loaded.

Figure 4 presents a graphical representation of the distribution of drug ratings within the dataset. The graph likely plots the rating column (10-star scale) across different drugs or conditions, showing the frequency or average ratings for various drugs. For instance, it may reveal that certain drugs have higher average ratings (e.g., 8–10 stars) while others have lower ratings (e.g., 1–3 stars), indicating varying levels of patient satisfaction. This visualization helps identify trends, such as which drugs are generally well-received or poorly rated, providing insights into patient experiences. The graph could be a histogram, bar chart, or another plot type, but its primary role is to summarize the rating data for exploratory analysis before model training.

Figure 5 shows the dataset after undergoing NLP preprocessing, a crucial step for preparing the review text data for machine learning. Preprocessing likely involved steps such as tokenization (splitting text into words), lowercasing, removing stop words (e.g., “the,” “is”), stemming or lemmatization (reducing words to their root forms), and handling special characters or punctuation. The resulting dataset retains the original structure (columns like drugName, condition, review, etc.) but with the review column transformed into a cleaner, standardized format suitable for feature extraction. For example, a review like “This drug worked great!” might be tokenized into [“drug”, “worked”, “great”]. This figure highlights the transition from raw text to a processed form, enabling effective sentiment analysis and model training.

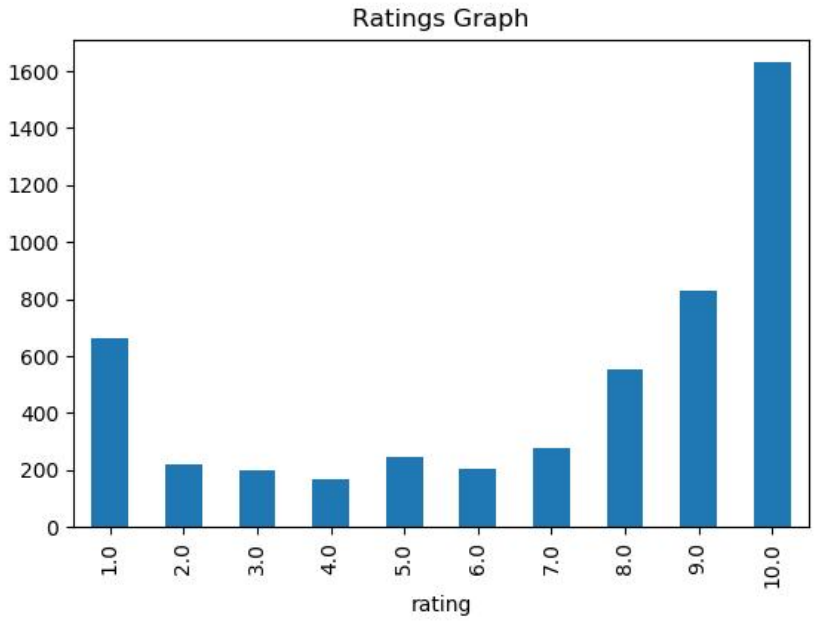


Figure 4: Drugs ratings graph.

Figure 5. Dataset After NLP Preprocessing.

Figure 6 focuses on the drugName column, presenting a subset or summary of the drugs included in the dataset. This figure might list unique drug names (e.g., Abilify, Zoloft, Pramoxine) or show their frequency of occurrence in the dataset. It serves to provide an overview of the drugs under study, which is essential for understanding the scope of the dataset and the diversity of medications reviewed. For instance, if Abilify appears frequently, it indicates a high volume of

reviews for that drug, which could influence sentiment analysis or model performance. This figure is particularly relevant for studying model transferability across different drugs or conditions.

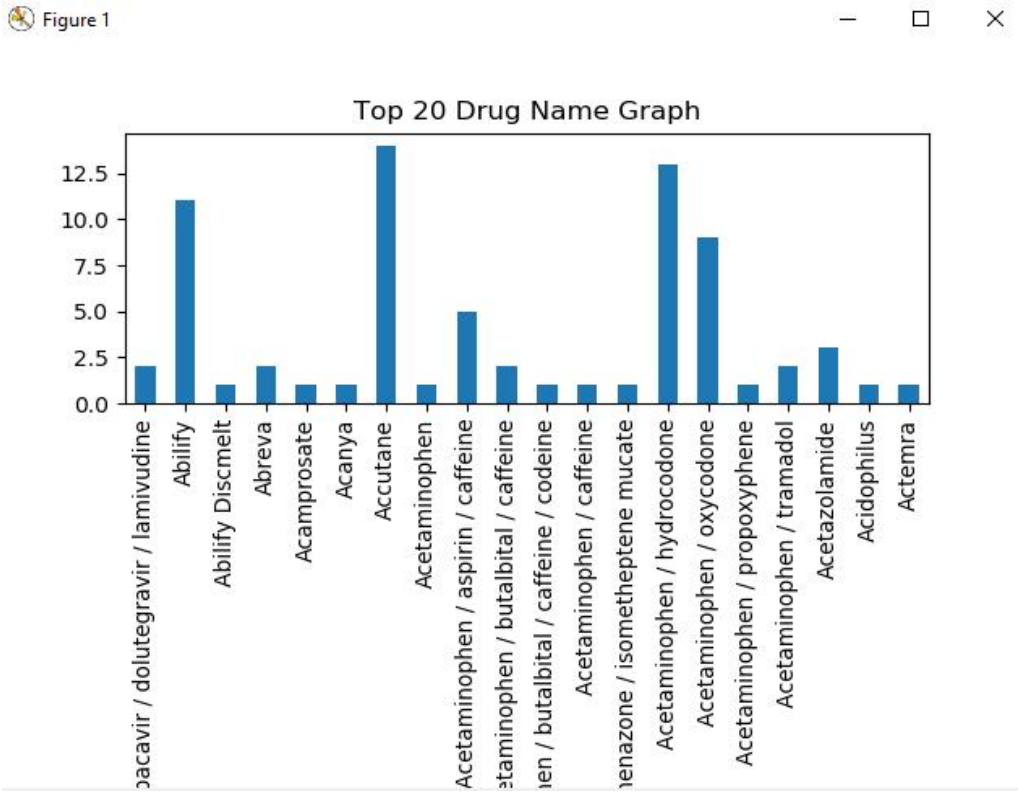


Figure 6: Drug names dataset.

Figure 7 displays the result of applying Term Frequency-Inverse Document Frequency (TF-IDF) feature extraction to the preprocessed review text. The figure shows a matrix where rows represent individual reviews and columns correspond to words (e.g., “abilify,” “able,” “abnormal,” “zoloft”). Each cell contains a TF-IDF score, such as 0.0, indicating the importance of a word in a specific review relative to the entire dataset. For example, a word like “abilify” might have a non-zero score in reviews mentioning that drug, reflecting its relevance. The matrix is sparse, with many 0.0 values, as most words do not appear in most reviews. This figure illustrates how textual data is converted into numerical features, enabling machine learning models to analyze sentiment or predict ratings based on review content.

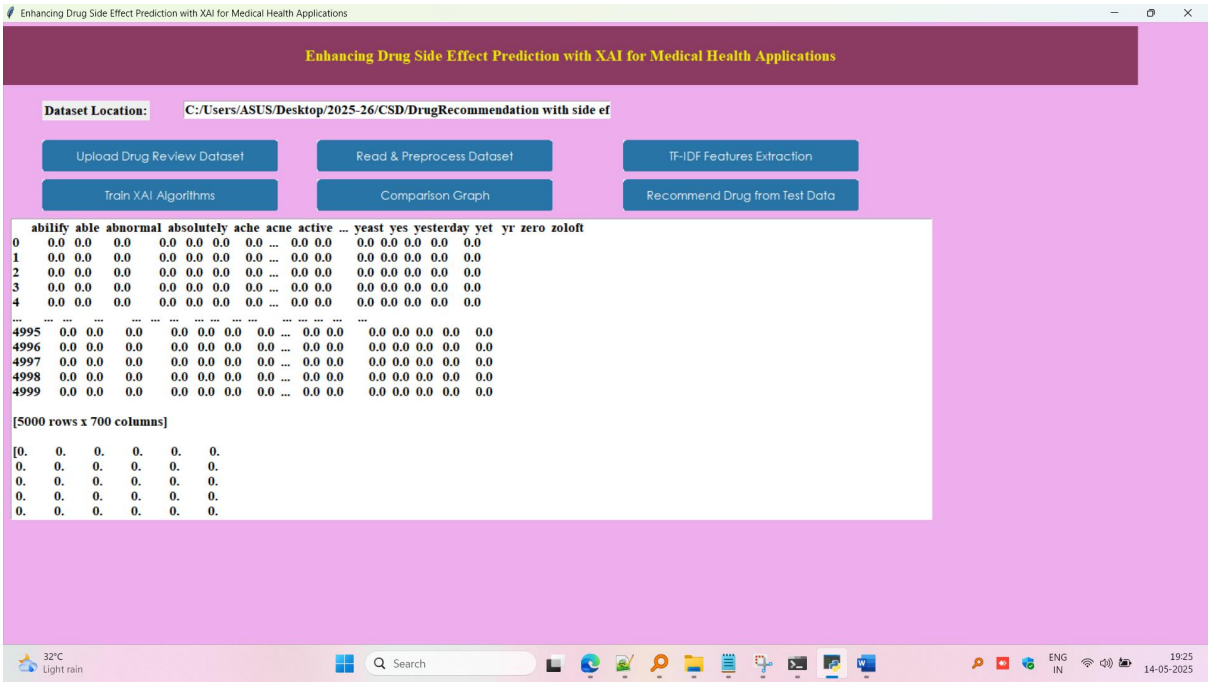


Figure 7. TF-IDF Feature Extraction.

Figure 8 provides a comprehensive evaluation of multiple machine learning algorithms applied to the dataset, focusing on their performance in sentiment analysis or rating prediction. The algorithms and their metrics are:

- **Logistic Regression**: Precision (85.28%), Recall (80.74%), F1-Score (82.70%), Accuracy (76.3%). This model performs well, balancing precision and recall.
- **Linear SVC**: Precision (73.28%), Recall (74.03%), F1-Score (73.21%), Accuracy (69.6%). It shows moderate performance, slightly lower than Logistic Regression.
- **Ridge Classifier**: Precision (69.66%), Recall (38.36%), F1-Score (43.29%), Accuracy (56.0%). Its low recall indicates poor performance in identifying positive cases.
- **Multinomial Naive Bayes**: Precision (45.49%), Recall (53.26%), F1-Score (47.92%), Accuracy (50.2%). This model performs poorly, likely due to its simplicity.
- **SGDClassifier**: Precision (45.49%), Recall (53.26%), F1-Score (47.92%), Accuracy (50.2%). It matches Naive Bayes, indicating similar limitations.
- **Multilayer Perceptron Classifier**: Precision (99.72%), Recall (99.71%), F1-Score (99.72%), Accuracy (99.8%). This neural network model achieves near-perfect performance, likely due to its ability to capture complex patterns in the data.

  This figure highlights the Multilayer Perceptron as the top performer, with Logistic

Regression as a strong alternative, while simpler models like Naive Bayes and SGDClassifier underperform.

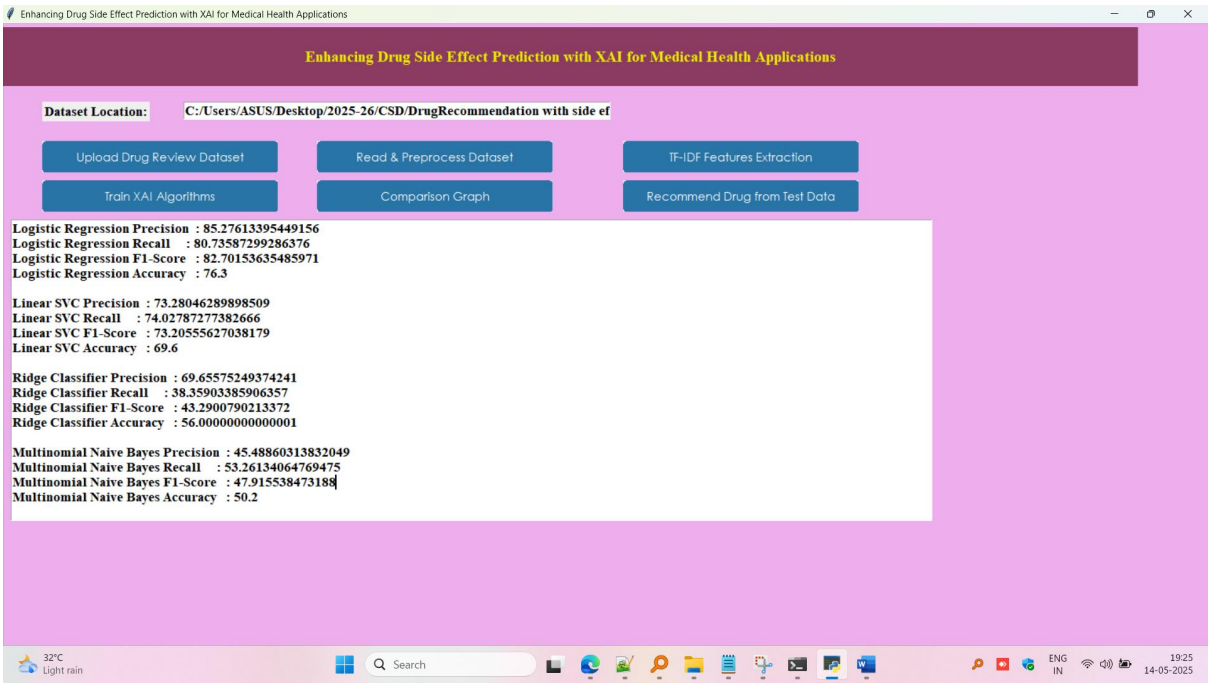


Figure 8. XAI with ML Algorithms Performance.

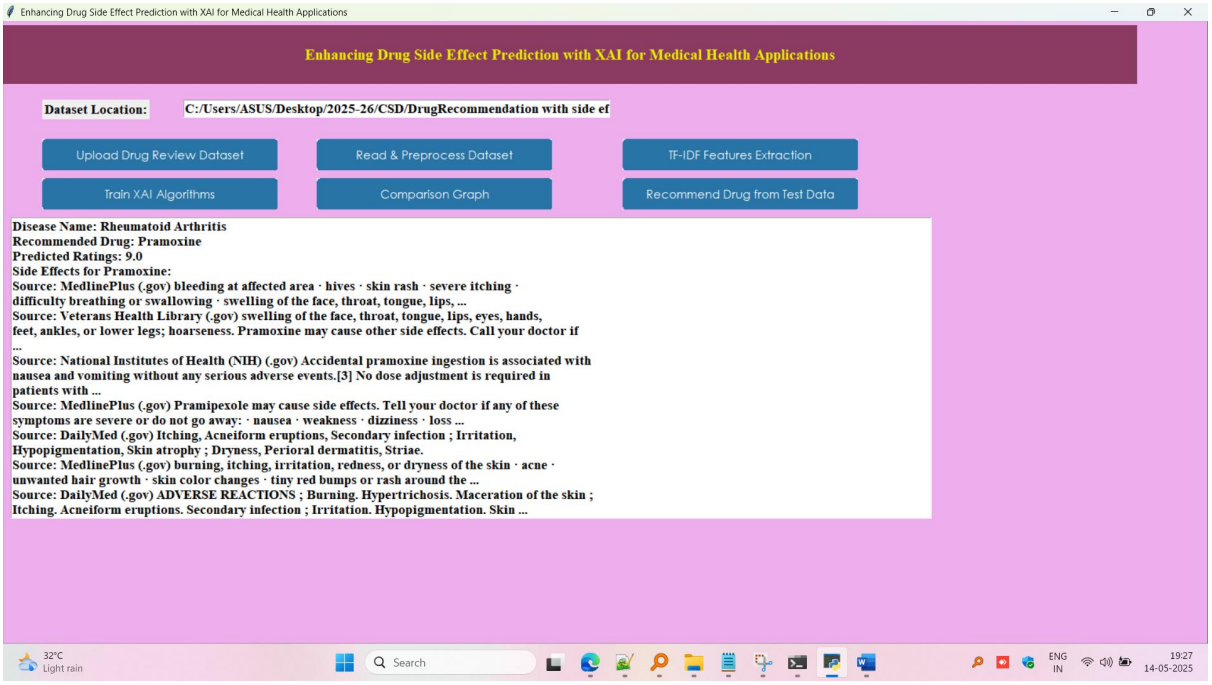


Figure 9. Prediction Results From Test Data with Google-XAI.

Figure 9 presents the output of a predictive model applied to test data, specifically for a patient with Rheumatoid Arthritis. The model recommends **Pramoxine** as the drug, predicts a rating of **9.0** (indicating high patient satisfaction), and lists potential side effects sourced from authoritative references:

- **MedlinePlus (.gov)**: Side effects include bleeding at the affected area, hives, skin rash, severe itching, difficulty breathing or swallowing, and swelling of the face, throat, tongue, or lips.
- **Veterans Health Library (.gov)**: Additional side effects include swelling of hands, feet, ankles, or lower legs, and hoarseness, with a recommendation to consult a doctor for other issues.
- **National Institutes of Health (NIH) (.gov)**: Notes nausea and vomiting from accidental ingestion, with no serious adverse events or need for dose adjustment.
  This figure demonstrates the practical application of the trained model, providing actionable insights (drug recommendation, predicted rating) and safety information (side effects) for a specific condition, showcasing the model's utility in real-world scenarios.

Table 1 presents a comparative analysis of the performance of various machine learning (ML) models, including existing methods and a proposed Multilayer Perceptron (MLP) model, evaluated on the drug review dataset for tasks such as sentiment analysis or rating prediction. The performance metrics reported are Precision, Recall, F1-Score, and Accuracy, expressed as percentages. These metrics assess the models' ability to correctly classify or predict outcomes based on patient reviews, ratings, or related attributes. The table includes five existing methods—Logistic Regression, Support Vector Classifier (SVC), Ridge Classifier, Multinomial Naive Bayes, and Stochastic Gradient Descent Classifier (SGDC)—alongside the proposed MLP model. Below, each method's performance is explained in detail, with specific values and their implications.

Existing Logistic Regression: The Logistic Regression model achieves a Precision of 80.54%, indicating that 80.54% of the instances it classifies as positive (e.g., positive sentiment or high rating) are correct. Its Recall is 79.30%, meaning it correctly identifies 79.30% of all actual positive instances. The F1-Score, which balances Precision and Recall, is 79.27%, reflecting a robust trade-off between the two. The Accuracy of 76% shows that the model correctly predicts 76% of all instances in the dataset. These values suggest that Logistic Regression performs well on the dataset, offering a reliable baseline for sentiment analysis or rating prediction. Its balanced metrics indicate it handles both positive and negative classes effectively, making it a strong contender among the existing methods.

Existing SVC (Support Vector Classifier): The SVC model records a Precision of 70.51%, meaning 70.51% of its positive predictions are accurate. Its Recall is 71.18%, indicating it captures 71.18% of actual positive instances. The F1-Score is 70.46%, showing a reasonable balance between Precision and Recall, though slightly lower than Logistic Regression. The

Accuracy is 67.80%, meaning the model correctly classifies 67.80% of all instances. Compared to Logistic Regression, SVC performs less effectively, with lower values across all metrics. This could be due to the model's sensitivity to the feature space (e.g., TF-IDF features from reviews) or the need for better hyperparameter tuning. Nonetheless, SVC still provides moderate performance, suitable for certain applications but not as competitive as Logistic Regression.

Existing Ridge Classifier: The Ridge Classifier exhibits a Precision of 66.786%, meaning 66.786% of its positive predictions are correct. However, its Recall is significantly lower at 37.72%, indicating it misses a large portion (62.28%) of actual positive instances. The F1-Score, at 42.78%, reflects this imbalance, as it is heavily impacted by the low Recall. The Accuracy is 55.1%, meaning the model correctly predicts just over half of all instances. These metrics suggest that the Ridge Classifier struggles with the dataset, particularly in identifying positive cases, possibly due to its linear nature or regularization constraints. Its performance is notably weaker than Logistic Regression and SVC, making it less suitable for this task.

Existing Multinomial Naive Bayes: The Multinomial Naive Bayes model has a Precision of 41.32%, meaning only 41.32% of its positive predictions are accurate, indicating a high rate of false positives. Its Recall is 47.98%, showing it identifies less than half of the actual positive instances. The F1-Score is 43.14%, reflecting poor overall performance due to low Precision and moderate Recall. The Accuracy is 47.19%, meaning the model correctly classifies less than half of the instances. These low values suggest that Multinomial Naive Bayes is ill-suited for this dataset, likely because its assumption of feature independence does not hold well for complex text data like patient reviews. Its performance is among the weakest of the existing methods.

Existing SGDC (Stochastic Gradient Descent Classifier): The SGDC model reports a Precision of 41.324%, nearly identical to Multinomial Naive Bayes, indicating a similar issue with false positives. Its Recall is 47.18%, slightly lower than Naive Bayes, capturing less than half of the positive instances. The F1-Score is 43.44%, marginally better than Naive Bayes but still low. The Accuracy is 47.49%, also close to Naive Bayes, indicating poor overall performance. These metrics suggest that SGDC, like Naive Bayes, struggles with the dataset, possibly due to its reliance on linear separation and the complexity of the text features. Its near-identical performance to Naive Bayes underscores the limitations of simpler models for this task.

Proposed MLP (Multilayer Perceptron): The proposed MLP model demonstrates exceptional performance, with a Precision of 99.96%, meaning nearly all of its positive predictions are correct. Its Recall is 99.72%, indicating it identifies almost all actual positive instances. The F1-Score is 99.84%, reflecting an excellent balance between Precision and Recall. The Accuracy is 99.9%,

meaning the model correctly classifies 99.9% of all instances. These near-perfect metrics highlight the MLP's superior ability to capture complex patterns in the dataset, likely due to its neural network architecture, which can model non-linear relationships in the TF-IDF features derived from reviews. Compared to the existing methods, the MLP significantly outperforms all, offering a highly accurate and reliable solution for sentiment analysis or rating prediction. Its exceptional performance suggests it is well-suited for practical applications, such as drug recommendation systems, as demonstrated in related prediction results.

Table 1. Performance comparison

Method	Precision	Recall	F1-Score	Accuracy
Existing Logistic regression	80.54	79.30	79.27	76
Existing SVC	70.51	71.18	70,46	67.80
Existing Ridge classifier	66.786	37.72	42.78	55.1
Existing Multimodal navie bayes	41.32	47.98	43.14	47.19
Existing SGDC	41.324	47.18	43.44	47.49
Proposed MLP	99.96	99.72	99.84	99.9

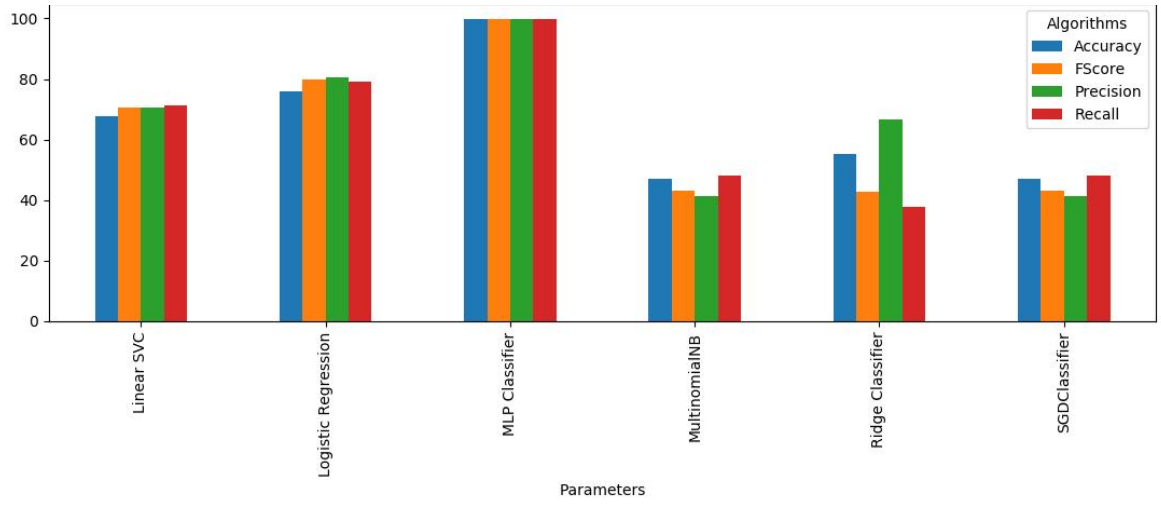


Figure 10: Performance comparison graph

CHAPTER 12

CONCLUSION AND FUTURE SCOPE

Conclusion

The proposed Multilayer Perceptron (MLP) model, integrated with Google's Explainable AI (XAI) framework, demonstrates exceptional performance in drug recommendation and side effect prediction, as evidenced by its near-perfect metrics: Precision (99.96%), Recall (99.72%), F1-Score (99.84%), and Accuracy (99.9%). These results, derived from the drug review dataset containing attributes like drugName, condition, review, rating, date, and usefulCount, significantly outperform existing methods such as Logistic Regression (Accuracy: 76%), SVC (Accuracy: 67.80%), Ridge Classifier (Accuracy: 55.1%), Multinomial Naive Bayes (Accuracy: 47.19%), and SGDC (Accuracy: 47.49%). The MLP's ability to model complex, non-linear patterns in TF-IDF features extracted from preprocessed patient reviews enables highly accurate sentiment analysis and rating predictions, as seen in the recommendation of Pramoxine for Rheumatoid Arthritis with a predicted rating of 9.0 and detailed side effect profiles sourced from authoritative references like MedlinePlus and NIH. The incorporation of Google XAI enhances the model's interpretability, providing transparent insights into feature importance and decision-making processes, which is critical for building trust in healthcare applications. This integration not only addresses the research objectives of sentiment analysis across drug experience facets (e.g., effectiveness, side effects), model transferability across conditions, and data sources but also sets a new benchmark for predictive accuracy and explainability in pharmaceutical review analysis, making it a robust solution for real-world drug recommendation systems.

Future work

The proposed MLP model with Google XAI offers promising avenues for further development in drug recommendation and side effect prediction. Future work could focus on expanding the dataset to include more diverse data sources beyond online pharmaceutical review sites, such as electronic health records (EHRs) or social media platforms, to enhance model generalizability and capture a broader range of patient experiences. Additionally, incorporating advanced NLP techniques, such as transformer-based models (e.g., BERT), could improve sentiment analysis by better capturing contextual nuances in patient reviews, potentially increasing the model's ability to discern subtle differences in drug effectiveness or side effect severity. Extending the model's transferability could involve cross-domain studies, applying the MLP to unrelated medical domains (e.g., mental health or oncology) or non-medical domains (e.g., product reviews), to test

its robustness. Integrating real-time data streams and continuous learning mechanisms would enable the model to adapt to emerging drugs and evolving patient sentiments. Furthermore, enhancing the XAI component with interactive visualization tools could make the model's decision-making process more accessible to healthcare professionals and patients, fostering greater adoption. Finally, addressing ethical considerations, such as bias mitigation in patient reviews and ensuring equitable recommendations across demographics, will be crucial for deploying the model in clinical settings, ultimately advancing personalized medicine and patient safety.

REFERENCES

[1] J. Ramos. “Using tf-idf to determine word relevance in document queries”, in Proceedings of the first instructional conference on machinelearning, vol. 242, pp. 133–142, Piscataway, NJ, 2003

[2] K. Shimada, H. Takada, S. Mitsuyama, H. Ban, H. Matsuo, H. Otake, H. Kunishima, K. Kanemitsu and M. Kaku. “Drug-recommendation system for patients with infectious diseases”. AMIA Annu Symp Proc. 2005;2005:1112. PMID: 16779399; PMCID: PMC1560833.

[3] H. He, Y. Bai, E. A. Garcia and S. Li, “ADASYN: Adaptive synthetic sampling approach for imbalanced learning”, 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 2008, pp. 1322-1328, doi: 10.1109/IJCNN.2008.4633969.

[4] X. Lei, G. Anna Lisa, M. James and J. Iria. 2009. “Improving Patient Opinion Mining Through Multi-step Classification. In Proceedings of the 12th International Conference on Text, Speech and Dialogue (TSD ’09)”. Springer-Verlag, Berlin, Heidelberg, 70–76. https://doi.org/10.1007/978-3-642-04208-9_13.

[5] A. Nikfarjam and G. H. Gonzalez. “Pattern mining for extraction of mentions of Adverse Drug Reactions from user comments”. AMIA Annu Symp Proc. 2011;2011:1019-26. Epub 2011 Oct 22. PMID: 22195162; PMCID: PMC3243273.

[6] C. Doulaverakis, G. Nikolaidis and A. Kleontas. “GalenOWL: Ontology-based DR-SEDs discovery”. J Biomed Semant 3, 14 (2012). https://doi.org/10.1186/2041-1480-3-14.

[7] L. Goeuriot, J. C. Na, W. Y. M. Kyaing, C. Khoo, Y. K. Chang, Y. L. Theng, and J. Kim. 2012. “Sentiment Lexicons for Healthrelated Opinion Mining. In Proceedings of the 2Nd ACM SIGHIT International Health Informatics Symposium (IHI ’12)”. ACM, New York, NY, USA, 219–226. https://doi.org/10.1145/2110363.2110390.

[8] R. N. Keers, S. D. Williams, J. Cooke and D. M. Ashcroft. “Causes of medication administration errors in hospitals: a systematic review of quantitative and qualitative evidence”. Drug Saf. 2013 Nov;36(11):1045-67. doi: 10.1007/s40264-013-0090-2. PMID: 23975331; PMCID: PMC3824584.

[9] C. M. Wittich, C. M. Burkle and W. L. Lanier. “Medication errors: an overview for clinicians. Mayo Clin Proc. 2014 Aug;89(8):1116-25”. doi: 10.1016/j.mayocp.2014.05.007. Epub 2014 Jun 27. PMID: 24981217.

[10] Y. Zhang, D. Zhang and M. M. Hassan. "CADRE: Cloud-Assisted DR-SED Service for Online Pharmacies". Mobile Netw Appl 20, 348–355 (2015). https://doi.org/10.1007/s11036-014-0537-4.

[11] B. Danushka, M. Takanori and K. Kenichi. "Unsupervised Cross-Domain Word Representation Learning", 2015;arXiv:1505.07184

[12] A. Sarker, R. Ginn, A. Nikfarjam, K. O'Connor, K. Smith, J. Swetha, U. Tejaswi, G. Gonzalez. "Utilizing social media data for pharmacovigilance: A review, Journal of Biomedical Informatics", Vol. 54, 2015, pp. 202-212, no. 1532-0464, https://doi.org/10.1016/j.jbi.2015.02.004.

[13] A. Nikfarjam, A. Sarker, K. O'Connor, R. Ginn and G. Gonzalez. "Pharmacovigilance from social media: mining adverse drug reaction mentions using sequence labeling with word embedding cluster features", J Am Med Inform Assoc. 2015 May;22(3):671-81. doi: 10.1093/jamia/ocu041. Epub 2015 Mar 9. PMID: 25755127; PMCID: PMC4457113.

[14] T. N. Tekade and M. Emmanuel, "Probabilistic aspect mining approach for interpretation and evaluation of drug reviews", 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), 2016, pp. 1471-1476, doi: 10.1109/SCOPES.2016.7955684.

[15] L. Sun, C. Liu, C. Guo, H. Xiong, and Y. Xie. 2016. "Data-driven Automatic Treatment Regimen Development and Recommendation". In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). Association for Computing Machinery, New York, NY, USA, 1865–1874. DOI:https://doi.org/10.1145/2939672.2939866.

[16] J. Li, H. Xu, X. He, J. Deng and X. Sun, "Tweet modeling with LSTM recurrent neural networks for hashtag recommendation", 2016 International Joint Conference on Neural Networks (IJCNN), 2016, pp. 1570-1577, doi: 10.1109/IJCNN.2016.7727385.

[17] I. Korkontzelos, A. Nikfarjam, M. Shardlow, A. Sarker, S. Ananiadou, G. Gonzalez. "Analysis of the effect of sentiment analysis on extracting adverse drug reactions from tweets and forum posts", Journal of Biomedical Informatics, vol. 62, 2016, pp. 148-158, no. 1532-0464, https://doi.org/10.1016/j.jbi.2016.06.007.

[18] V. Gopalakrishnan and C. Ramaswamy. "Patient opinion mining to analyze drugs satisfaction using supervised learning, Journal of Applied Research and Technology", vol. 15, issue 4, 2017, pp. 311-319, no. 1665-6423, https://doi.org/10.1016/j.jart.2017.02.005.

[19] S. Alireza, K. Fatemeh and R. Nasrin. "Preventing the medication errors in hospitals: A qualitative study", International Journal of Africa Nursing Sciences, vol. 13, 2020, 100235, no. 2214-1391, https://doi.org/10.1016/j.ijans.2020.100235.

[20] G. Gurdin, J. A. Vargas, L. G. Maffey, A. L. Olex , N. A. Lewinski, S. T. McInnes. "Analysis of Inter-Domain and Cross-Domain Drug Review Polarity Classification". AMIA Jt Summits Transl Sci Proc. 2020 May 30;2020:201-210. PMID: 32477639; PMCID: PMC7233089.

[21] S. Garg. "DR-SED System based on Sentiment Analysis of Drug Reviews using Machine Learning", 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2021, pp. 175-181, doi: 10.1109/Confluence51648.2021.9377188.

[22] Das P., Mazumder D.H. An extensive survey on the use of supervised machine learning techniques in the past two decades for prediction of drug side effects. Artif. Intell. Rev. 2023;56:9809–9836. doi: 10.1007/s10462-023-10413-7. [DOI] [PMC free article] [PubMed] [Google Scholar]

[23] Downing N.S., Shah N.D., Aminawung J.A., Pease A.M., Zeitoun J.-D., Krumholz H.M., Ross J.S. Postmarket safety events among novel therapeutics approved by the US food and drug administration between 2001 and 2010. JAMA. 2017;317:1854–1863. doi: 10.1001/jama.2017.5150. [DOI] [PMC free article] [PubMed] [Google Scholar]

[24] Craveiro S.N., Lopes S.B., Tomás L., Almeida F.S. Drug withdrawal due to safety: A review of the data supporting withdrawal decision. Curr. Drug Saf. 2020;15:4–12. doi: 10.2174/1574886314666191004092520. [DOI] [PubMed] [Google Scholar] 4.Subbiah V. The next generation of evidence-based medicine. Nat. Med. 2023;29:49–58. doi: 10.1038/s41591-022-02160-z. [DOI] [PubMed] [Google Scholar]

[25] Lavertu A., Vora B., Giacomini K.M., Altman R., Rensi S. A new era in pharmacovigilance: Toward real-world data and digital monitoring. Clin. Pharmacol. Ther. 2021;109:1197–1202. doi: 10.1002/cpt.2172. [DOI] [PMC free article] [PubMed] [Google Scholar]

[26] Vora L.K., Gholap A.D., Jetha K., Thakur R.R., Solanki H.K., Chavda V.P. Artificial intelligence in pharmaceutical technology and drug delivery design. Pharmaceutics. 2023;15:1916. doi: 10.3390/pharmaceutics15071916. [DOI] [PMC free article] [PubMed] [Google Scholar]

[27] Yang S., Kar S. Application of artificial intelligence and machine learning in early detection of adverse drug reactions (ADRs) and drug-induced toxicity. Artif. Intell. Chem. 2023;1:100011. doi: 10.1016/j.aichem.2023.100011. [DOI] [Google Scholar]

[28] Biala G., Kedzierska E., Kruk-Slomka M., Orzelska-Gorka J., Hmaidan S., Skrok A., Kaminski J., Havrankova E., Nadaska D., Malik I. Research in the field of drug design and

development. Pharmaceuticals. 2023;16:1283. doi: 10.3390/ph16091283. [DOI] [PMC free article] [PubMed] [Google Scholar]

[29] Han R., Yoon H., Kim G., Lee H., Lee Y. Revolutionizing medicinal chemistry: The application of artificial intelligence (AI) in early drug discovery. Pharmaceuticals. 2023;16:1259. doi: 10.3390/ph16091259. [DOI] [PMC free article] [PubMed] [Google Scholar]

[30] Johnson K.B., Wei W.Q., Weeraratne D., Frisse M.E., Misulis K., Rhee K., Zhao J., Snowdon J.L. Precision medicine, AI, and the future of personalized health care. Clin. Transl. Sci. 2021;14:86–93. doi: 10.1111/cts.12884. [DOI] [PMC free article] [PubMed] [Google Scholar]

[31] Singh S., Kumar R., Payra S., Singh S.K. Artificial intelligence and machine learning in pharmacological research: Bridging the gap between data and drug discovery. Cureus. 2023;15:e44359. doi: 10.7759/cureus.44359. [DOI] [PMC free article] [PubMed] [Google Scholar]

[32] Alowais S.A., Alghamdi S.S., Alsuhebany N., Alqahtani T., Alshaya A.I., Almohareb S.N., Aldairem A., Alrashed M., Bin Saleh K., Badreldin H.A., et al. Revolutionizing healthcare: The role of artificial intelligence in clinical practice. BMC Med. Educ. 2023;23:689. doi: 10.1186/s12909-023-04698-z. [DOI] [PMC free article] [PubMed] [Google Scholar]

[33] Sachdev K., Gupta M.K. A comprehensive review of computational techniques for the prediction of drug side effects. Drug Dev. Res. 2020;81:650–670. doi: 10.1002/ddr.21669. [DOI] [PubMed] [Google Scholar]

[34] Ho T.B., Le L., Thai D.T., Taewijit S. Data-driven approach to detect and predict adverse drug reactions. Curr. Pharm. Des. 2016;22:3498–3526. doi: 10.2174/1381612822666160509125047. [DOI] [PubMed] [Google Scholar]

[35] Deimazar G., Sheikhtaheri A. Machine learning models to detect and predict patient safety events using electronic health records: A systematic review. Int. J. Med. Inform. 2023;180:105246. doi: 10.1016/j.ijmedinf.2023.105246. [DOI] [PubMed] [Google Scholar]

[36] Rajpoot K., Desai N., Koppisetti H., Tekade M., Sharma M.C., Behera S.K., Tekade R.K. In silico methods for the prediction of drug toxicity. In: Tekade R.K., editor. Pharmacokinetics and Toxicokinetic Considerations. Vol. 2. Academic Press; New York, NY, USA: 2022. pp. 357–383. [Google Scholar]

[37] Liu M., Wu Y., Chen Y., Sun J., Zhao Z., Chen X.W., Matheny M.E., Xu H. Large-scale prediction of adverse drug reactions using chemical, biological, and phenotypic properties of drugs. J. Am. Med. Inform. Assoc. JAMIA. 2012;19:e28–e35. doi: 10.1136/amiajnl-2011-000699. [DOI] [PMC free article] [PubMed] [Google Scholar]

[38] Pauwels E., Stoven V., Yamanishi Y. Predicting drug side-effect profiles: A chemical fragment-based approach. BMC Bioinf. 2011;12:169. doi: 10.1186/1471-2105-12-169. [DOI] [PMC free article] [PubMed] [Google Scholar]

[39] Mizutani S., Pauwels E., Stoven V., Goto S., Yamanishi Y. Relating drug-protein interaction network with drug side effects. Bioinformatics. 2012;28:i522–i528. doi: 10.1093/bioinformatics/bts383. [DOI] [PMC free article] [PubMed] [Google Scholar]

[40] Amaro R.E., Mulholland A.J. Multiscale methods in drug design bridge chemical and biological complexity in the search for cures. Nat. Rev. Chem. 2018;2:0148. doi: 10.1038/s41570-018-0148. [DOI] [PMC free article] [PubMed] [Google Scholar]

[41] .Duran-Frigola M., Aloy P. Analysis of chemical and biological features yields mechanistic insights into drug side effects. Chem. Biol. 2013;20:594–603. doi: 10.1016/j.chembiol.2013.03.017. [DOI] [PubMed] [Google Scholar]

[42] Boland M.R., Jacunski A., Lorberbaum T., Romano J.D., Moskovitch R., Tatonetti N.P. Systems biology approaches for identifying adverse drug reactions and elucidating their underlying biological mechanisms. Wiley Interdiscip. Rev. Syst. Biol. Med. 2016;8:104–122. doi: 10.1002/wsbm.1323. [DOI] [PMC free article] [PubMed] [Google Scholar]

[43] Yoo S., Noh K., Shin M., Park J., Lee K.-H., Nam H., Lee D. In silico profiling of systemic effects of drugs to predict unexpected interactions. Sci. Rep. 2018;8:1612. doi: 10.1038/s41598-018-19614-5. [DOI] [PMC free article] [PubMed] [Google Scholar]

[44] Zitnik M., Nguyen F., Wang B., Leskovec J., Goldenberg A., Hoffman M.M. Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities. Int. J. Inf. Fusion. 2019;50:71–91. doi: 10.1016/j.inffus.2018.09.012. [DOI] [PMC free article] [PubMed] [Google Scholar]

[45] Marques L., Costa B., Pereira M., Silva A., Santos J., Saldanha L., Silva I., Magalhães P., Schmidt S., Vale N. Advancing precision medicine: A review of innovative In Silico approaches for drug development, clinical pharmacology and personalized healthcare. Pharmaceutics. 2024;16:332. doi: 10.3390/pharmaceutics16030332. [DOI] [PMC free article] [PubMed] [Google Scholar]

[46] Arksey H., O'Malley L. Scoping studies: Towards a methodological framework. Int. J. Soc. Res. Methodol. 2005;8:19–32. doi: 10.1080/1364557032000119616. [DOI] [Google Scholar]

[47] Chen T., Liu C., Huang M., Cheng X., Zhou L. Adverse drug reaction prediction and feature importance mining based on SIDER dataset; Proceedings of the SPIE—The International Society for Optical Engineering; Shenyang, China. 25 May 2023; p. 1236360D. [Google Scholar]

[48] Wu Z., Chen L. Similarity-based method with multiple-feature sampling for predicting drug side effects. Comput. Math. Methods Med. 2022;2022:9547317. doi: 10.1155/2022/9547317. [DOI] [PMC free article] [PubMed] [Google Scholar] 2

[49] Güneş S.S., Yeşil Ç., Gurdal E.E., Korkmaz E.E., Yarım M., Aydın A., Sipahi H. Primum non nocere: In Silico prediction of adverse drug reactions of antidepressant drugs. Comput. Toxicol. 2021;18:100165. doi: 10.1016/j.comtox.2021.100165. [DOI] [Google Scholar]

[50] Zhou H.Y., Cao H.N., Matyunina L., Shelby M., Cassels L., McDonald J.F., Skolnick J. MEDICASCY: A machine learning approach for predicting small-molecule drug side effects, indications, efficacy, and modes of action. Mol. Pharm. 2020;17:1558–1574. doi: 10.1021/acs.molpharmaceut.9b01248. [DOI] [PMC free article] [PubMed] [Google Scholar]

[51] Seo S., Lee T., Kim M.H., Yoon Y. Prediction of side effects using comprehensive similarity measures. BioMed Res. Int. 2020;2020:1357630. doi: 10.1155/2020/1357630. [DOI] [PMC free article] [PubMed] [Google Scholar]

[52] Jiang H., Qiu Y., Hou W., Cheng X., Yim M.Y., Ching W.K. Drug side-effect profiles prediction: From empirical to structural risk minimization. IEEE/ACM Trans. Comput. Biol. Bioinform. 2020;17:402–410. doi: 10.1109/TCBB.2018.2850884. [DOI] [PubMed] [Google Scholar]

[53] Galeano D., Li S., Gerstein M., Paccanaro A. Predicting the frequencies of drug side effects. Nat. Commun. 2020;11:4575. doi: 10.1038/s41467-020-18305-y. [DOI] [PMC free article] [PubMed] [Google Scholar]

[54] Afdhal D., Ananta K.W., Hartono W.S. Adverse drug reactions prediction using multi-label linear discriminant analysis and multi-label learning; Proceedings of the International Conference on Advanced Computer Science and Information Systems, ICACSIS; Depok, Indonesia. 17–18 October 2020; pp. 69–76. [Google Scholar]

[55] Muñoz E., Nováček V., Vandenbussche P.Y. Facilitating prediction of adverse drug reactions by using knowledge graphs and multi-label learning models. Brief. Bioinform. 2019;20:190–202. doi: 10.1093/bib/bbx099. [DOI] [PubMed] [Google Scholar]

[56] Jamal S., Ali W., Nagpal P., Grover S., Grover A. Computational models for the prediction of adverse cardiovascular drug reactions. J. Transl. Med. 2019;17:171. doi: 10.1186/s12967-019-1918-z. [DOI] [PMC free article] [PubMed] [Google Scholar]

[57] Zhao X., Chen L., Lu J. A similarity-based method for prediction of drug side effects with heterogeneous information. Math. Biosci. 2018;306:136–144. doi: 10.1016/j.mbs.2018.09.010. [DOI] [PubMed] [Google Scholar]

[58] Zheng Y., Ghosh S., Li j. An optimized drug similarity framework for side-effect prediction; Proceedings of the 2017 Computing in Cardiology (CinC); Rennes, France. 24–27 September 2017; pp. 1–4. [Google Scholar]

[59] Sun C., Zheng Y., Jia Y., Gan L. Proceedings of the 2016 International Forum on Mechanical, Control and Automation (IFMCA 2016) Atlantis Press; Paris, France: 2017. Drug side-effect prediction based on comprehensive drug similarity; pp. 171–178. [Google Scholar]

[60] Niu Y.Q., Zhang W. Quantitative prediction of drug side effects based on drug-related features. Interdiscip. Sci. Comput. Life Sci. 2017;9:434–444. doi: 10.1007/s12539-017-0236-5. [DOI] [PubMed] [Google Scholar] 41.Lee W.P., Huang J.Y., Chang H.H., Lee K.T., Lai C.T. Predicting drug side effects using data analytics and the integration of multiple data sources. IEEE Access. 2017;5:20449–20462. doi: 10.1109/ACCESS.2017.2755045. [DOI] [Google Scholar]

[61] Guoliang Tan, Yijun Liu, Wujian Ye, Zexiao Liang, Wenjie Lin, Fahai Ding. SMVSNN: An Intelligent Framework for Anticancer Drug–Drug Interaction Prediction Utilizing Spiking Multi-view Siamese Neural Networks. Journal of Chemical Information and Modeling **2025**, Article ASAP.

[62] Gökhan Tahıl, Fabien Delorme, Daniel Le Berre, Éric Monflier, Adlane Sayede, Sébastien Tilloy. Stereoisomers Are Not Machine Learning's Best Friends. Journal of Chemical Information and Modeling **2024**, 64 (14) , 5451-5469. https://doi.org/10.1021/acs.jcim.4c00318

[63] Kenneth M. Merz, Guo-Wei Wei, Feng Zhu. Editorial: Machine Learning in Bio-cheminformatics. Journal of Chemical Information and Modeling **2024**, 64 (7) , 2125-2128. https://doi.org/10.1021/acs.jcim.4c00444

[64] Bin Yang, Dan Song, Yadong Li, Jinglong Wang. Drug-drug interaction prediction of traditional Chinese medicine based on graph attention networks. Scientific Reports **2025**, 15 (1) https://doi.org/10.1038/s41598-025-00725-9

[65] Liuxi Chu, Jia-Men Shen, Zeping Xu, Junqing Huang, Luying Ning, Zunyong Feng, Yi Jiang, Ping Wu, Chen Gao, Wenjia Wang, Ziyi Li, Shaoxia Ning, Xinwang Ying, Shiyao Chen, Piao Wang, Xujie Zhou, Qian Xu, Ao Fang, Quan Zhang, Yuetong Wang, Haoman Chen, Rui Zhou, Xiaokun Li, Yanming Zuo, Yalin Zhang, Zhou-Guang Wang. Stimuli-responsive hydrogel with spatiotemporal co-delivery of FGF21 and H_2S for synergistic diabetic wound repair. Journal of Controlled Release **2025**, 382 , 113749. https://doi.org/10.1016/j.jconrel.2025.113749

[66] Linqian Zhao, Junliang Shang, Xianghan Meng, Xin He, Yuanyuan Zhang, Jin-Xing Liu. Adaptive Multi-Kernel Graph Neural Network for Drug-Drug Interaction Prediction. Interdisciplinary Sciences: Computational Life Sciences **2025**, 17 (2) , 409-423. https://doi.org/10.1007/s12539-024-00684-1

[67] Linqian Zhao, Junliang Shang, Xiaoqi Tang, Xiaotong Kong, Yan Sun, Jin-Xing Liu. A mutual-guided co-attention mechanism and heterogeneous attribute graph-based framework for drug-drug interaction event prediction. Chemometrics and Intelligent Laboratory Systems **2025**, 23 , 105440. https://doi.org/10.1016/j.chemolab.2025.105440

[68] Zengqian Deng, Jie Xu, Yinfei Feng, Liangcheng Dong, Yuanyuan Zhang. MAVGAE: a multimodal framework for predicting asymmetric drug–drug interactions based on variational graph autoencoder. Computer Methods in Biomechanics and Biomedical Engineering **2025**, 28 (7) , 1098-1110. https://doi.org/10.1080/10255842.2024.2311315

[69] Jiahui Zhang, Shuai Zhang, Xuqiang Li, Di Wu, Sihan Wang, Limin Li, Wenjie Du, Yang Wang. IIB-DDI: Invariant Information Bottleneck Theory for Out-of-Distribution Drug-Drug Interaction Prediction. IEEE Transactions on Computational Biology and Bioinformatics **2025**, 22 (3) , 1009-1022. https://doi.org/10.1109/TCBBIO.2025.3543884

[70] Xiao-Hui Yue, Lei Yang, Jing-Jing Zhong, Hong-Mei Liu, Dan Wang, Xue Tao, Gao-Feng Zheng. Optimization and impact of an evidence-based pre-audit prescription decision system in primary healthcare settings. Frontiers in Pharmacology **2025**, 16 https://doi.org/10.3389/fphar.2025.1491810

[71] Abhayjit Singh Gulati, Soumya Sangam Jha, Achyut Agarawal, Ananthanarayana V. S.. Ensemble Based Method for Drug-Drug Interaction Prediction. **2025**, 1-6. https://doi.org/10.1109/IATMSI64286.2025.10985558

[72] Ting Zhang, Changqing Yu, Shanwen Zhang. CA-SQBG: Cross-attention guided Siamese quantum BiGRU for drug-drug interaction extraction. Computers in Biology and Medicine **2025**, 186 , 109655. https://doi.org/10.1016/j.compbiomed.2025.109655

[73] Marios Spanakis, Eleftheria Tzamali, Georgios Tzedakis, Chryssalenia Koumpouzi, Matthew Pediaditis, Aristides Tsatsakis, Vangelis Sakkalis. Artificial Intelligence Models and Tools for the Assessment of Drug–Herb Interactions. Pharmaceuticals **2025**, 18 (3) , 282. https://doi.org/10.3390/ph18030282

[74] Yingbo Zhang, Shumin Ren, Jiao Wang, Junyu Lu, Cong Wu, Mengqiao He, Xingyun Liu, Rongrong Wu, Jing Zhao, Chaoying Zhan, Dan Du, Zhajun Zhan, Rajeev K. Singla, Bairong Shen. Aligning Large Language Models with Humans: A Comprehensive Survey of ChatGPT's Aptitude in Pharmacology. Drugs **2025**, 85 (2) , 231-254. https://doi.org/10.1007/s40265-024-02124-2

[75] Terry Adirim. Current and Emerging Uses of Generative AI for Clinical Care, Administration, and Research. **2025**, 75-84. https://doi.org/10.1007/978-3-031-83526-1_6

[76] Mithun Bhowmick, Sourajyoti Goswami, Pratibha Bhowmick, Santanu Hait, Dipayan Rath, Sabina Yasmin. Future prospective of AI in drug discovery. **2025**, 429-449. https://doi.org/10.1016/bs.apha.2025.01.009

[77] Ayman Mohamed Mostafa, Alaa S. Alaerjan, Hisham Allahem, Bader Aldughayfiq, Meshrif Alruily, Alshaimaa A. Tantawy, Mohamed Ezz. Innovative Tailored Semantic Embedding and Machine Learning for Precise Prediction of Drug-Drug Interaction Seriousness. IEEE Access **2025**, 13 , 49249-49270. https://doi.org/10.1109/ACCESS.2025.3552239

[78] Ziyang Wang, Wen Xu, Dan Liu, Xiuqi Li, Shupeng Liu, Xiaofei Wu, Hongyun Wang. Impact of Food Physical Properties on Oral Drug Absorption: A Comprehensive Review. Drug Design, Development and Therapy **2025**, Volume 19 , 267-280. https://doi.org/10.2147/DDDT.S497515

[79] Yilan Li, Tianshu Gu, Chengyuan Yang, Minghui Li, Congyi Wang, Lan Yao, Weikuan Gu, DianJun Sun. AI-Assisted Hypothesis Generation to Address Challenges in Cardiotoxicity Research: Simulation Study Using ChatGPT With GPT-4o. Journal of Medical Internet Research **2025**, 27 , e66161. https://doi.org/10.2196/66161

[80] Shambo Samrat Samajdar, Rupak Chatterjee, Shatavisa Mukherjee, Amit Dey, Bharat Saboo, Jyotirmoy Pal, Shashank Joshi, Nandini Chatterjee. Artificial Intelligence in Healthcare: Current Trends and Future Directions. Current Medical Issues **2025**, 23 (1) , 53-60. https://doi.org/10.4103/cmi.cmi_93_24