

OsteoNet: AI-Driven Osteoporosis Prediction Using Integrated Imaging and Numeric Data

Project Report

submitted in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING
(ARTIFICIAL INTELLIGENCE and DATA SCIENCE)

By

R. MONALISHA	(21341A4550)
P. PAVAN KUMAR	(21341A4543)
B. BALAJI	(21341A4506)
K. SANKAR RAO	(21341A4527)
P. ANUSHA	(21341A4546)

Under the Guidance of
Mr. D. Siva Krishna

Assistant Professor
Department of CSE

GMR Institute of Technology

An Autonomous Institute Affiliated to JNTU-GV, Vizianagaram
(Accredited by NBA, NAAC with 'A' Grade & ISO 9001:2015 Certified Institution)

GMR Nagar, Rajam – 532127,
Andhra Pradesh, India
May 2024

CERTIFICATE

This is to certify that the project report entitled “**OsteoNet: AI-Driven Osteoporosis Prediction Using Integrated Imaging and Numeric Data**” is the bonafide record of project work carried out under my supervision by **R. MONALISHA (21341A4550), P. PAVAN KUMAR (21341A4543), B. BALAJI (21341A4506), K. SANKAR RAO (21341A4527), and P. ANUSHA (21341A4546)**, during the academic year 2021-2025, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering in Artificial Intelligence and Data Science of Jawaharlal Nehru Technological University, Vizianagaram. The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

Signature of Project Guide

Mr. D. Siva Krishna

Assistant Professor
Department of CSE
GMRIT

Head of the Department

Dr. K. Srividya

Associate Professor and Head
Department of CSE (AI & DS)
GMRIT

The report is submitted for the viva-voce examination held on

Signature of Internal Examiner

Signature of External Examiner

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to acknowledge the assistance and cooperation we have received from several persons while undertaking this B. Tech. Mini Project. We owe special debt of gratitude to **Mr. D. Siva Krishna** Department of Computer Science & Engineering, for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us.

We also take the opportunity to acknowledge the contribution of **Dr. K. Srividya**, Head, Department of CSE (Artificial Intelligence and Data Science), for her full support and assistance during the development of the project.

We would like to take this opportunity to thank our beloved Principal **Dr. C. L. V. R. S. V. Prasad**, for providing all the necessary facilities and a great support to us in completing the project work

We would like to thank all the faculty members of the Department of CSE (Artificial Intelligence and Data Science) for their direct or indirect support and all the lab technicians for their valuable suggestions and for providing excellent opportunities in the completion of this report.

R. MONALISHA (21341A4550)

P. PAVAN KUMAR (21341A4543)

B. BALAJI (21341A4506)

K. SANKAR RAO (21341A4527)

P. ANUSHA (21341A4546)

ABSTRACT

Knee osteoporosis, characterized by bone weakening and increased fragility leading to fractures, affects millions in India and globally. In India, approximately six crore individuals suffer from osteoporosis, with women constituting 80% of cases, and onset occurring earlier compared to Western countries. Globally, osteoporosis prevalence stands at approximately 18.3%, disproportionately affecting women at 23.1%. Additionally, one in three women and one in five men over the age of 50 are affected by this condition. This project aims to enhance the accuracy of knee osteoporosis prediction through a deep learning approach integrating image and numerical data. Leveraging advanced machine learning techniques, our research seeks to improve early detection and intervention strategies, potentially alleviating the burden on individuals and health care systems. We propose utilizing deep learning technology to automate Bone Mineral Density (BMD) measurements, transfer learning from pre-trained neural networks to create reliable models, employing joint learning frameworks to enhance diagnostic accuracy, improving image processing and feature extraction using convolutional neural networks (CNN), and developing methods to differentiate between normal, osteopenia and osteoporosis bone tissue. Deep learning techniques hold immense potential to revolutionize osteoporosis diagnosis by improving accuracy, accessibility, and efficiency, thereby aiding in the early detection and management of this prevalent condition.

Keywords: Knee osteoporosis, deep learning, X-ray analysis, ensemble approach, diagnostic accuracy.

CONTENTS

Page No.

<i>Acknowledgement</i>	<i>i</i>
<i>Abstract</i>	<i>ii</i>
<i>Table of figures</i>	<i>iv</i>
<i>Table of tables</i>	<i>v</i>
Chapter 1: INTRODUCTION	1
Chapter 2: LITERATURE REVIEW	2
Chapter 3: METHODOLOGY	5
3.1 DATASET	5
3.2 MACHINE LEARNING ALGORITHMS	6
3.3 DEEP LEARNING ALGORITHMS	11
Chapter 4: WORK FLOW	18
4.1 FUSION MODEL	19
Chapter 5: CODE IMPLEMENTATION	20
5.1 MACHINE LEARNING ALGORITHMS	20
5.2 DEEP LEARNING ALGORITHMS	24
5.3 FUSION MODEL	34
5.4 FLASK INTEGRATION	40
Chapter 6: RESULTS AND DISCUSSIONS	47
6.1 DISCUSSION	47
6.2 WEB INTERFACE DESIGN	48
6.3 RESULT DISPLAY	51
6.4 TECHNICAL DETAILS	51
Chapter 7: CONCLUSION	52

List of Figures

Fig-1:	Three Stages of Osteoporosis	5
Fig-2:	Data Preparation	6
Fig-3:	Decision Tree for Classification	7
Fig-4:	A Decision Tree from Random Forest for Classification	8
Fig-5:	Support vector machine (SVM) for Classification	9
Fig-6:	VGG16 for Training and Validation Loss	12
Fig-7:	Alex Net for Training and Validation Loss	16
Fig-8:	Architecture of Alex Net	17
Fig-9:	Dataflow diagram	18
Fig-10:	Final model for Training and validation Loss and accuracy	19
Fig-11:	Home Page	49
Fig-12:	Login and Registration Page	49
Fig-13:	Form page	50
Fig-14:	Storing Registered User details in Database	50
Fig-15:	Final Result	51

List of Tables

	Page No.
Table1: Performance of Machine Learning Models	47
Table2: Performance of Deep Learning Models	47
Table3: Performance of Fusion Model	48

1. INTRODUCTION

Osteoporosis is a condition characterized by the gradual weakening of bones over time, often without any warning signs until fractures occur, typically affecting areas such as the hip, wrist, spine, elbow, pelvis, knee, or shoulder. Osteoporosis poses significant danger as it renders bones fragile and prone to fractures even from minor stresses like bending or coughing. Early detection is paramount, yet challenging due to the lack of symptoms in the initial stages. Particularly vulnerable are women, especially those in older age groups post-menopause. In India alone, approximately 50 million people grapple with osteoporosis or low bone mass, underscoring its significance as a healthcare concern. This condition affects individuals of all genders and age groups, with women and older adults being more susceptible, as evidenced by approximately 200 million affected women globally. Diagnosis relies heavily on Bone Mineral Density tests (DXA), though accessibility can be limited in certain regions. Treatment costs vary widely, from affordable generics to expensive brand names, adding complexity to management strategies. The heightened fragility of bones increases susceptibility to fractures even from minor accidents, resulting in pain and disability. Severe bone loss can alter one's physical appearance, contributing to height loss and a stooped posture. Comprehensive approaches to detection, treatment, and management are imperative to mitigate the impact of osteoporosis on both individuals and communities. Identifying patients with osteoporosis presents challenges due to the absence of early symptoms, and doctors may encounter difficulties in diagnosis. Complications include noticeable changes in spine shape, leading to pain and disability. Challenges in identification primarily arise from the asymptomatic nature of early stages, emphasizing the need for regular screening and awareness campaigns.

2. LITERATURE SURVEY

Osteoporosis affects various segments of society, including patients, their families, government agencies, and medical institutions. So many Researchers analysed data from 86 studies involving 103,334,579 individuals aged 15–105 years. The global prevalence of osteoporosis was reported to be 18.3%. Among all the men affected in worldwide is 11.7%, the Highest prevalence in Africa is 39.5%, Lowest prevalence in Netherlands is 4.1% and Among women worldwide is 23.1%. To estimate and summarize the prevalence of osteoporosis among adults in India. The prevalence of osteoporosis was significantly higher among females 26.3% and Males is 10.9%. Osteoporosis is 22.9% and Osteopenia is 44.8% [1]. The age-standardized prevalence of osteoporosis in China was estimated to be 33.49% Men is 20.73% and Women is 38.05%. The factors such as low BMI, and current regular smoking were significantly associated with a higher risk of osteoporosis and osteopenia in the middle-aged and elderly population.[2]. Dual-energy X-ray absorptiometry (DXA) is a medical imaging technique used to measure bone mineral density (BMD). This is the most widely used method for assessing fracture risk, particularly in postmenopausal women. Challenges with DXA: T-score limitations, Ignores Bone size and Consequences are Unnecessary treatment for those with low fracture risk.[3]. BMD introduces support vector machines (SVM) and regression trees as the chosen machine learning techniques to analyse the relationship between bone mineral density (BMD) and various factors such as diet and lifestyle.[4].GP kernels enhance osteoporosis risk factor identification, emphasizing rigorous statistical analysis. The study addresses limitations in traditional methods for identifying osteoporosis risk factors, proposing a more accurate and robust diagnostic approach. The approach combines rigorous statistical analysis and integrated GP kernels to enhance intractability and capture complex data relationships, improving accuracy. The study falls short in detailing specific limitations or potential drawbacks associated with the proposed method, leaving key information undisclosed. Study likely assesses accuracy, sensitivity, specificity, AUC for osteoporosis diagnosis.[5]. The study addresses timely osteoporosis detection in RA using DCNNs and explores nursing care's impact on self-efficacy for improved chronic condition management outcomes. Deep convolutional neural network (DCNN) is a type of artificial intelligence model specifically designed for analysing image data. DCNNs improve accuracy, efficiency in osteoporosis detection, adopting a holistic approach. Study lacks DCNN details, training data size, self-efficacy methods [8]. CNNs excel in image analysis like X-ray interpretation, autonomously extracting features without manual intervention's excel in osteoporosis diagnosis, promoting automation, cost-effectiveness, and early detection. Model performance depends on data quality/quantity. Deep learning challenges interpretability, raising

transparency, generalizability, and ethical concerns, including bias.[9]. DXA underutilized due to cost and accessibility constraints in osteoporosis assessment. CT enhances osteoporosis detection for at-risk individuals without DXA Computational models optimize procedures, aiding surgeons in personalized approach selection. Finite Element Analysis simulates structural behaviour under diverse loads.[10]. The researchers developed a deep learning model that was trained on a dataset of hip radiographs and corresponding clinical data from patients. The article investigated the effectiveness of five different convolutional neural network (CNN) models for classifying osteoporosis is VGG16, VGG19, Alex Net, GoogleNet, ResNet-50. The ensemble models, which combined each CNN model with patient clinical data.[12]. CNNs excel in image analysis like X-ray interpretation, autonomously extracting features without manual intervention. The model performance data quality and quantity and includes interpretability, raising transparency, generalizability, and ethical concerns, including bias.[13]. Study likely focuses on CT -to-DXA calibration methods for Mbiti can QCT enhances osteoporosis detection for at-risk individuals without DXA and CT -to -BMD accuracy impacted by various factors.[14]. Computational models optimize procedures, aiding surgeons in personalized approach selection and Finite Element Analysis (FEA) provides detailed insights into stress, load transfer, and implant risks.[15]. The study explores in deep learning knee osteoporosis and osteoarthritis and it improves the accuracy, efficiency, and reduced subjectivity in diagnosis and it have lack of architectures and training data.[16]. Here they used five different convolutional neural network (CNN) models for classifying osteoporosis those are GoogleNet, EfficientNet B0, EfficientNet B3, ResNet-18 and ResNet-50 and the ensemble model with EfficientNet B3 and clinical data achieved the highest accuracy.[17].

Osteoporosis Prediction and Detection (OPAD) it could incorporate various functionalitye for detecting osteoporosis.it can based on various functionalities like patient Data Collection, Fracture Risk Assessment, DXA Scan Recommendation and Treatment Recommendations and can reduce fracture risk within a short time frame.[18]. Ensemble deep learning models combined with panoramic radio graphs and Convolutional Neural Networks (CNNs) are used like EfficientNet-b0, -b3, -b7, ResNet-18, -50, and -152. it can be conducted on a relatively small dataset of 778 patients.[19]. Osteoporosis is suggests using CT scans instead of X-rays for better results and CT scans better and introduces a new method called MVCTNet that works well for diagnosing osteoporosis. This method includes some Challenges that are not knowing much about the pictures they used, and the methods might

not work as well with other types of images.[20]. OPAD standardizes medical decision-making by providing instant diagnostic comments, 10-year fracture risk assessment, treatment options. It gives accurate in fracture risk evaluation, further cost-benefit studies are needed to determine OPAD's cost-

effectiveness in fracture liaison services.[21]. clinical decision support system for the diagnosis, fracture risks, and treatment of osteoporosis, which can provide valuable assistance to healthcare professionals in managing this condition.[22]. The feasibility of T1 and T2 imaging of knee cartilage at 7T in comparison to 3T and determined the ability of T1 r and T2 to differentiate between normal and osteoarthritis (OA) patients. 7T imaging provides increased signal-to-noise ratio (SNR) compared to 3T, resulting in improved image quality and 7T requires large amounts of radio frequency (RF) energy, which has raised concerns about specific absorption rate.[23]. Multimodal and multiparametric imaging techniques have been optimized and combined to assess complementary aspects of osseous metastases. Multimodal imaging techniques allow for a comprehensive assessment of bone metastases, combining the advantages of different modalities.[24]. Quantitative computed tomography (QCT) serves as an alternative method for bone mineral density (BMD) testing when access to dual-energy X-ray absorption (DXA) is restricted and QCT can accurately identify unsuspected osteoporotic compression fractures, enabling osteoporosis diagnosis independent of DXA T-score. CT has lower precision compared to DXA, but it can still be adequate for monitoring longitudinal changes in BMD.[25]. The multi-user CT network (MVCTNet) captures various features from the images, improving the performance of the classification task and dataset used for evaluation contains CT images from 2,883 patients, but the representative and diversity of the dataset are not discussed.[26]. The study uses data from a single centre, potentially affecting the generalizability of the model to other populations and healthcare settings and Transfer learning can help to improve the accuracy of models trained on small datasets.[27]. osteoporosis with a focus on addressing gaps in understanding the condition and improving assessment and treatment for high-risk individuals. veganism, non-coeliac gluten-free diets affect bone mineral density through microbiome-mediated pathways. It Research into the role of diet in bone health has traditionally focused on calcium and vitamin D. [28]. Osteoporosis recommends resistance exercise combining 3 to 10 types of free weight and mechanical exercise. The major muscle groups, performed with an intensity of 50% to 85% 1-repetition maximum, 5 to 12 repetitions/set, 2 to 3 days/week, for 3 to 12 months. Impact exercises such as jumping chin-ups with drop landings and jump rope are also recommended.[29]. An image processing technique called anisotropic Morlet wavelet transform to calculate the energy at different regions of the proximal femur. It was discovered that 21% (10/60), and 39% (23/60) of the survey Indian women were found to have osteoporosis and osteopenia separately.[30].

3. METHODOLOGY

3.1 DATASET

We started with a dataset sourced from Kaggle, containing both numerical and image data, all tied together through a common reference point. Initially, it comprised 239 data points, with 27 numerical attributes. One pivotal attribute was 'foreign', linking numerical data to respective images via Patient ID. Our dataset was divided into three final groups: Normal, Osteopenia, and Osteoporosis, with varying instances: 36 Normal, 154 Osteopenia, and 49 Osteoporosis cases. However, we noticed performance issues with our model, prompting a revaluation of our dataset.

To address this, we augmented our data, focusing on Normal and Osteoporosis cases, the primary areas of interest. We acquired an additional set of 186 Normal and 186 Osteoporosis images, albeit without accompanying text information. Merging this new data with the original dataset, we aimed for a balanced mix of images across categories.

After integration, our combined dataset swelled to 611 images. It included 36 Normal, 154 Osteopenia, and 49 Osteoporosis images from the original set, complemented by 222 Normal, Osteopenia 154 and 235 Osteoporosis images from the supplementary dataset. This consolidation ensured a more exhaustive dataset, enriching our analysis and bolstering model training and evaluation across all categories.



Fig-1: Three Stages of Osteoporosis

In essence, the integration of these datasets yielded a more comprehensive and balanced dataset. This comprehensive approach, leveraging both numerical and image data, promises enhanced model accuracy and robustness, setting the stage for improved predictive performance.

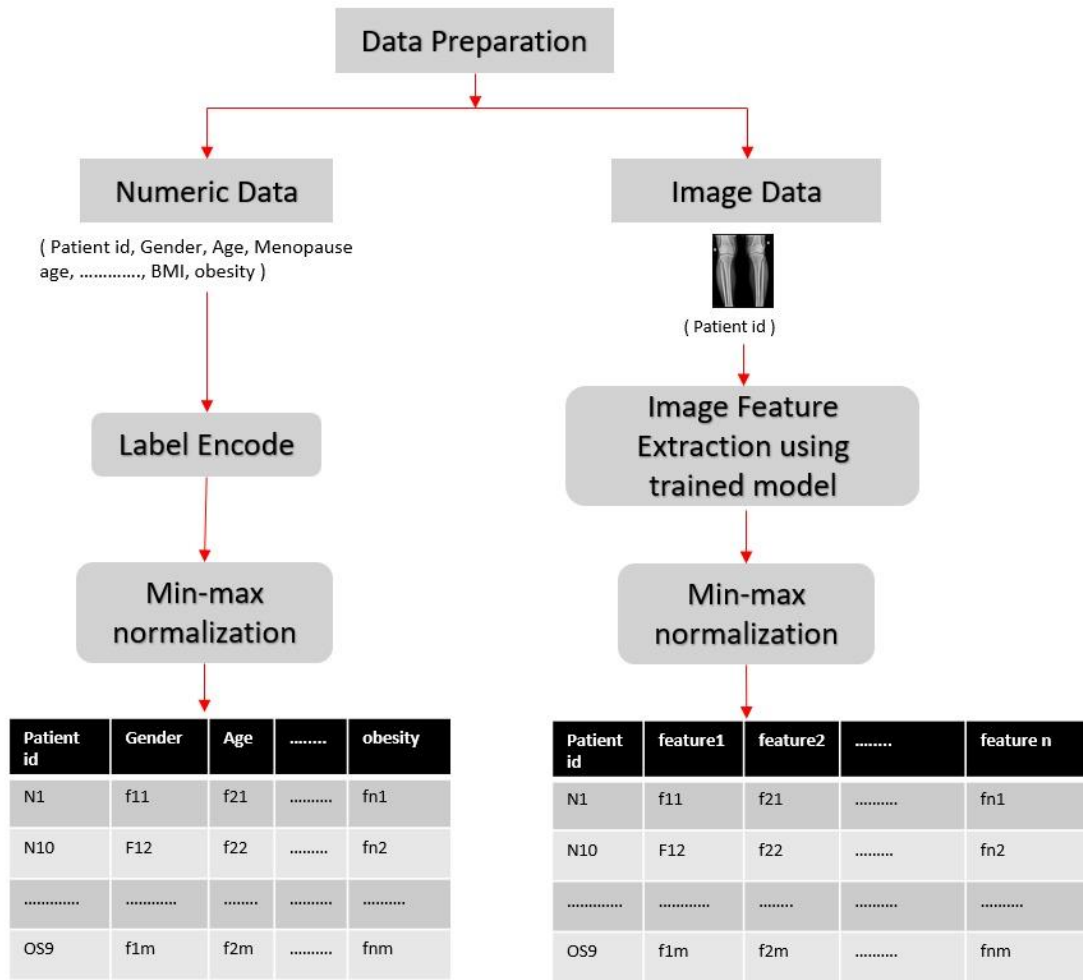


Fig-2: Data Preparation

3. 2 MACHINE LEARNING ALGORITHMS USED

Decision Tree

Decision Trees are hierarchical tree-like structures used for classification and regression tasks in machine learning. They work by recursively splitting the input space into regions based on the values of input features. At each step of the tree-building process, the algorithm selects the feature that best splits the data into homogeneous subsets, typically using metrics like Gini impurity or information gain.

In the context of textual data, Decision Trees can be applied by first converting the text data into numerical features using techniques like Bag-of-Words or TF-IDF. Bag-of-Words represents each document as a vector of word counts, while TF-IDF assigns weights to words based on their importance in the document and across the entire corpus.

When applied to textual data, Decision Trees can effectively learn decision rules based on the presence or absence of specific words or features in the text. This makes them particularly useful for tasks such as text classification or sentiment analysis, where the goal is to categorize text documents into predefined classes or categories.

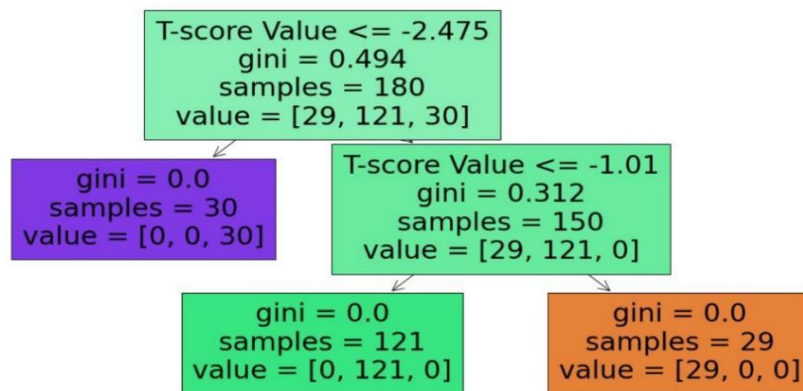


Fig-3: Decision Tree for Classification

One advantage of Decision Trees is their interpretability. Since the decision rules are represented as a tree structure, it is easy to understand and visualize the decision-making process. This makes Decision Trees suitable for tasks where model interpretability is important, such as in legal or regulatory domains.

However, Decision Trees are prone to overfitting, especially when dealing with high-dimensional data like text. To mitigate this issue, techniques like pruning or using ensemble methods like Random Forest can be employed.

Random Forest

Random Forest is an ensemble learning method that combines multiple Decision Trees to improve predictive performance and reduce overfitting. It works by training a collection of decision trees on random subsets of the training data and features and then averaging their predictions for classification tasks or taking a vote for regression tasks.

In the context of textual data, Random Forest can be applied similarly to Decision Trees by first converting the text data into numerical features. Random Forest introduces randomness in the tree-building process by selecting a random subset of features for each tree and bootstrap aggregating (bagging) to train each tree on a different subset of the training data.

Random Forest is well-suited for handling high-dimensional data like text because it reduces overfitting and increases robustness by aggregating the predictions of multiple trees. This makes it

particularly effective for tasks such as text classification, where the goal is to classify text documents into multiple categories or classes.

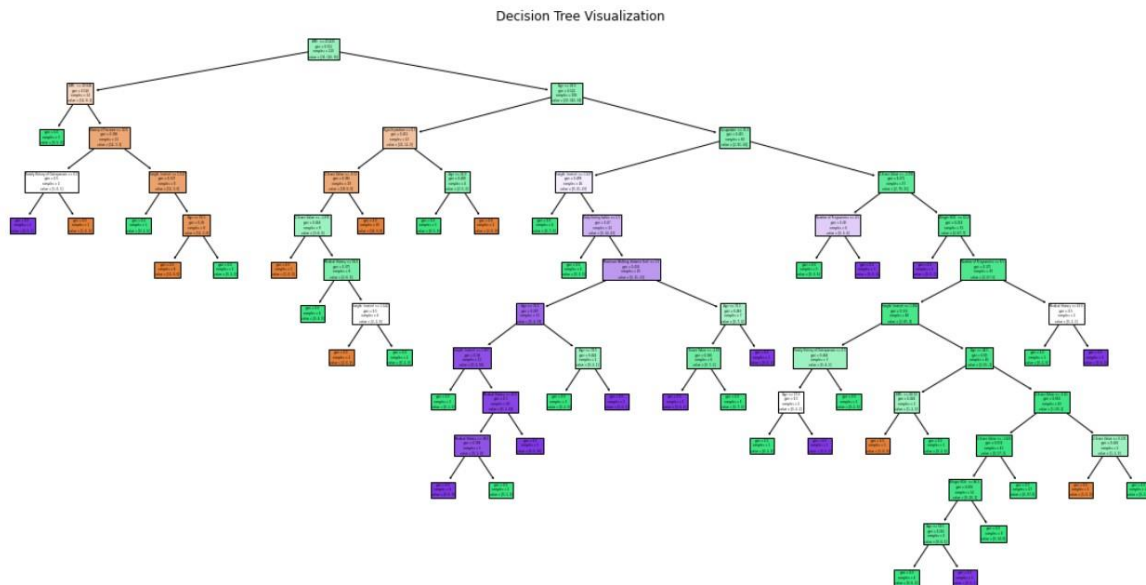


Fig-4: A Decision Tree from Random Forest for Classification

One of the key advantages of Random Forest is its ability to provide estimates of feature importance, which can help identify the most discriminative words or features in the text. This information can be valuable for understanding the underlying patterns in the data and improving model interpretability. However, Random Forests can be computationally expensive, especially when dealing with large datasets or a large number of trees. Additionally, they may not perform as well as more sophisticated algorithms like Support Vector Machines (SVM) on certain tasks.

Support Vector Machines (SVM)

Support Vector Machines (SVM) is a powerful supervised learning algorithm used for classification and regression tasks. It works by finding the hyperplane that best separates the data points of different classes in the feature space while maximizing the margin between the hyperplane and the nearest data points of each class.

In the context of textual data, SVM can be applied by first converting the text data into numerical features using techniques like Bag-of-Words, TF-IDF, or word embeddings. SVM aims to find the hyperplane that best separates the text documents belonging to different classes based on their feature representations.

SVM is well-suited for handling high-dimensional data like text because it can effectively deal with sparse feature spaces and nonlinear decision boundaries. By using the kernel trick, SVM can map the input features into a higher-dimensional space where the classes become separable by a hyperplane, even if they are not linearly separable in the original feature space.

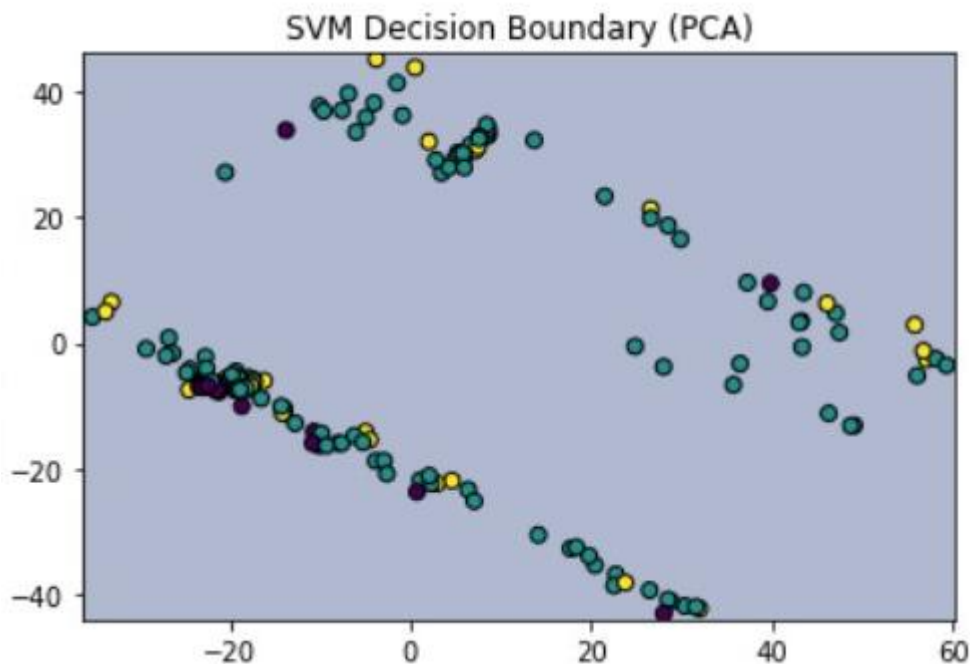


Fig-5: Support vector machine (SVM) for Classification

One of the advantages of SVM is its ability to control the complexity of the decision boundary through the choice of the kernel function and regularization parameters. This makes SVM highly adaptable to different types of textual data and can lead to better generalization performance compared to simpler algorithms like Decision Trees.

However, SVM can be sensitive to the choice of hyperparameters and the kernel function, which may require careful tuning for optimal performance. Additionally, SVMs can be computationally expensive, especially when dealing with large datasets or a large number of features.

In summary, Decision Trees, Random Forest, and Support Vector Machines are three popular machine learning algorithms for working with textual data. Each algorithm has its own strengths and weaknesses, and the choice of algorithm depends on factors such as the specific characteristics of the data, the task at hand, and computational considerations.

A COMPARISON OF CLASSIFICATION REPORTS OF MACHINE LEARNING

Classification Report:				
	precision	recall	f1-score	support
0	1.00	0.97	0.98	29
1	0.99	0.99	0.99	121
2	0.97	1.00	0.98	30
accuracy			0.99	180
macro avg	0.99	0.99	0.99	180
weighted avg	0.99	0.99	0.99	180

Decision Tree

Classification Report:				
	precision	recall	f1-score	support
0	0.89	1.00	0.94	8
1	1.00	0.97	0.98	33
2	1.00	1.00	1.00	19
accuracy			0.98	60
macro avg	0.96	0.99	0.98	60
weighted avg	0.99	0.98	0.98	60

Random Forest

Classification Report:				
	precision	recall	f1-score	support
0	0.89	1.00	0.94	8
1	0.97	0.97	0.97	33
2	0.94	0.89	0.92	19
accuracy			0.95	60
macro avg	0.93	0.95	0.94	60
weighted avg	0.95	0.95	0.95	60

SVM

The following attributes are typically involved in the above ML models, providing insights into the model's performance.

Precision - Precision measures the accuracy of positive predictions made by the model. It is calculated as the ratio of true positive predictions to the sum of true positives and false positives. Precision indicates how many of the instances predicted as positive are actually positive.

Recall (Sensitivity)- Recall measures the ability of the model to correctly identify positive instances out of all actual positive instances. It is calculated as the ratio of true positive predictions to the sum of true positives and false negatives. Recall indicates how many of the actual positive instances were correctly predicted by the model.

F1-Score- The F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall, giving equal weight to both metrics. F1-score is useful when you have an uneven class distribution.

Support- Support is the number of actual occurrences of each class in the test dataset. It represents the number of true instances in each class.

Accuracy- Accuracy measures the overall correctness of the model's predictions. It is calculated as the ratio of the number of correct predictions to the total number of predictions. Accuracy is a good measure when the classes are balanced.

Macro Average- The macro average calculates the average performance across all classes without considering class imbalance. It simply averages the precision, recall, and F1-score for all classes.

Weighted Average- The weighted average calculates the average performance across all classes, considering class imbalance. It is computed by weighting the contributions of each class by its support.

Confusion Matrix- The confusion matrix is a table that describes the performance of a classification model. It presents a summary of the model's predictions, showing the counts of true positives, true negatives, false positives, and false negatives.

These attributes collectively provide a comprehensive understanding of the classification model's performance, including its ability to correctly classify instances, handle class imbalances, and make overall accurate predictions.

3. 3 DEEP LEARNING ALGORITHMS USED

VGG16

VGG16 stands as a hallmark achievement in the landscape of deep learning, stemming from the collaborative efforts of the Visual Geometry Group (VGG) during the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Building upon the foundational principles of its predecessors, VGG16 embodies a profound leap forward with its architecture comprising 16 layers, notably deeper than its antecedents.

Central to VGG16's architecture is its stack of 13 convolutional layers followed by 3 fully connected layers, each layer accompanied by rectified linear unit (ReLU) activation functions. This design choice fosters efficient feature extraction, enabling the network to discern intricate patterns and structures within input images. The utilization of compact 3x3 convolutional filters and 2x2 max-pooling layers, with a stride of 2, underscores a uniform architecture, facilitating the extraction of spatial hierarchies of features. VGG16's significance extends beyond its architectural prowess; its pre-trained weights garnered from extensive training on the ImageNet dataset serve as invaluable assets in transfer learning scenarios. By leveraging the learned features encoded within VGG16, practitioners expedite model development and achieve superior performance on target tasks, especially in scenarios where labelled data is scarce.

Primarily tailored for image classification tasks, VGG16 exhibits exceptional accuracy in discerning objects and scenes within images. Its standardized architecture and adept feature extraction capabilities render it suitable for a spectrum of computer vision tasks, encompassing object detection, segmentation, and image retrieval.

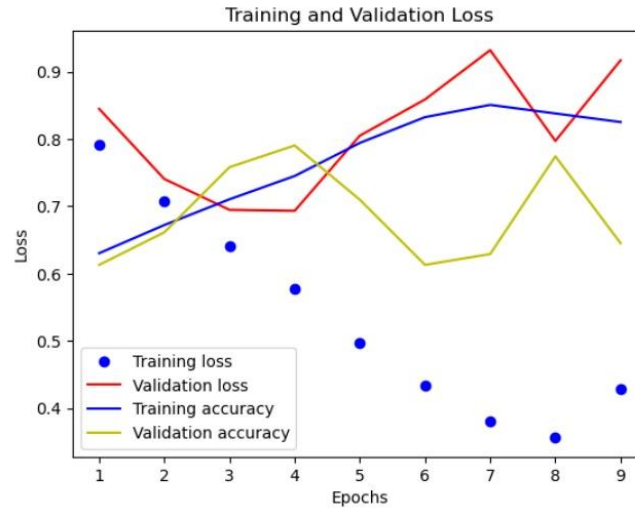


Fig-6: VGG16 for Training and Validation Loss

While VGG16's depth contributes to its ability to capture intricate features, it concurrently imposes computational demands. Nonetheless, the trade-off between computational resources and performance underscores its practical utility in various applications.

Moreover, VGG16's open-source nature fosters knowledge dissemination and collaboration within the machine learning community. Researchers and practitioners globally benefit from access to its implementation details, pre-trained models, and associated resources, thereby catalysing further advancements in computer vision research and application development.

In essence, VGG16 epitomizes a paradigm shift in deep learning, exemplifying the fusion of architectural sophistication and practical utility. Its standardized design, adept feature extraction capabilities, and pre-trained models establish it as a cornerstone in the realm of computer vision, empowering diverse applications and fostering continuous innovation.

VGG19

VGG19, an extension of the VGG16 architecture, emerged from the efforts of the Visual Geometry Group (VGG) during the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Upholding the fundamental principles of its predecessor, VGG19 presents a deeper network comprising 19 layers, thereby amplifying its capacity to discern intricate features from input images. Its architecture mirrors VGG16's with 16 convolutional layers followed by 3 fully connected layers, each accompanied by rectified linear unit (ReLU) activation functions, pivotal in feature extraction. The utilization of small 3x3 convolutional filters and 2x2 max-pooling layers, along with a stride of 2, perpetuates a uniform design, enriching the network's ability to grasp spatial hierarchies of features. VGG19's prowess in feature extraction remains a standout attribute, leveraging its profound architecture and pre-existing weights. In the realm of transfer learning, where models are fine-tuned

on specialized datasets with limited labelled data, VGG19's learned features serve as invaluable assets. By leveraging insights gleaned from pre-training on expansive datasets like ImageNet, practitioners expedite model development while achieving enhanced performance on target tasks.

Primarily tailored for image classification endeavors, VGG19 boasts exceptional accuracy in discerning objects and scenes within images. Its deep-seated architecture and standardized design render it apt for feature extraction across various computer vision domains, encompassing object detection, segmentation, and image retrieval. Furthermore, VGG19's role as a benchmark architecture fosters the evaluation of new CNN architectures and methodologies, thus propelling advancements in the realm of computer vision research.

While VGG19's extended depth augments its capacity for discerning intricate features, it concurrently escalates computational demands. However, the balance between performance and computational resources warrants consideration in practical deployments, with VGG19 typically offering superior accuracy albeit at an augmented computational expense.

The collaborative and open-source nature of VGG19 engenders a vibrant ecosystem of knowledge dissemination within the machine learning community. Researchers and practitioners worldwide benefit from access to the architecture's implementation specifics, pre-trained models, and associated resources, thereby catalyzing further innovations in computer vision research and application development.

In summation, VGG19 epitomizes an evolutionary stride beyond VGG16, emboldening its capacity to unravel intricate features from input images. Its uniform design, adept feature extraction capabilities, and pre-trained models position it as a pivotal asset across diverse computer vision tasks. Despite its heightened computational overhead, VGG19's indelible contributions to research and practical applications underscore its pivotal stature in the domain of computer vision.

ResNet50

ResNet50, an integral member of the ResNet (Residual Network) family, represents a seminal advancement in deep learning architecture, revolutionizing the realm of computer vision. Introduced by Kaiming He et al. in their 2015 paper "Deep Residual Learning for Image Recognition," ResNet50 embodies a profound departure from traditional convolutional neural network (CNN) architectures by introducing residual connections, which mitigate the vanishing gradient problem and enable the training of exceedingly deep networks.

At its core, ResNet50 comprises 50 layers, with a fundamental architectural innovation lying in the incorporation of residual blocks. These blocks circumvent the degradation problem encountered in

training extremely deep networks by introducing skip connections, also known as shortcut connections. These connections enable the network to learn residual mappings, effectively capturing the difference between the input and output of each block. Consequently, deeper ResNet variants like ResNet50 exhibit improved training convergence and scalability compared to their shallower counterparts.

The architecture of ResNet50 encompasses several stages, each consisting of convolutional layers, batch normalization, and rectified linear unit (ReLU) activation functions. Additionally, max-pooling layers are strategically placed to down sample feature maps, thereby capturing increasingly abstract representations of input images. Notably, ResNet50 employs bottleneck blocks in certain stages to reduce computational complexity while preserving representational capacity, a design choice crucial for achieving both efficiency and efficacy in deep learning models.

ResNet50's architectural innovations extend beyond training efficacy to encompass practical utility in transfer learning scenarios. Pre-trained on vast datasets like ImageNet, ResNet50 encodes rich feature representations that can be fine-tuned on specialized tasks with limited labelled data. This transfer learning capability accelerates model development and enhances performance across a spectrum of computer vision tasks, including image classification, object detection, and semantic segmentation.

The advent of ResNet50 has significantly influenced the landscape of deep learning research and application development. Its architectural elegance, coupled with its superior performance and transfer learning capabilities, has positioned it as a cornerstone in the domain of computer vision. Moreover, the open-source availability of pre-trained ResNet50 models and associated resources fosters collaboration and knowledge dissemination within the machine learning community, catalyzing further advancements in research and practical applications.

In summary, ResNet50 represents a paradigm shift in deep learning architecture, exemplifying the fusion of theoretical innovation and practical utility. Its residual connections, deep-seated architecture, and transfer learning capabilities establish it as a pivotal asset in addressing complex computer vision challenges and driving advancements in artificial intelligence.

InceptionNet

InceptionNet, also known as GoogLeNet, stands as a pioneering deep convolutional neural network (CNN) architecture developed by researchers at Google, particularly highlighted in the 2014 paper titled "Going Deeper with Convolutions" by Szegedy et al. InceptionNet introduces a novel concept

termed "Inception modules," which revolutionize the way feature maps are extracted and processed within the network.

At the heart of InceptionNet lies the Inception module, which comprises a set of parallel convolutional layers of different filter sizes (1x1, 3x3, 5x5), alongside max-pooling operations. By concurrently processing input feature maps through multiple pathways of varying receptive fields, Inception modules facilitate the extraction of multi-scale features, enabling the network to capture both fine-grained details and global context within images. Additionally, 1x1 convolutions are employed to reduce dimensionality and computational complexity, enhancing the efficiency of feature extraction. InceptionNet's architecture is characterized by its deep structure, featuring multiple stacked Inception modules interspersed with max-pooling layers for spatial down-sampling. Notably, the network culminates in global average pooling and a softmax layer for classification, eschewing traditional fully connected layers and reducing the number of parameters, thereby mitigating overfitting and improving generalization performance.

One of the key contributions of InceptionNet lies in its computational efficiency and parameter optimization. By leveraging the parallel processing capabilities of Inception modules and incorporating dimensionality reduction techniques, InceptionNet achieves competitive accuracy on image classification tasks while maintaining a relatively compact model size compared to contemporaneous architectures.

Furthermore, InceptionNet's architecture lends itself well to transfer learning, wherein pre-trained models on large-scale datasets like ImageNet can be fine-tuned on domain-specific tasks with limited labelled data. This transfer learning capability enhances model generalization and accelerates deployment across diverse applications in computer vision, including object recognition, detection, and segmentation.

InceptionNet's impact extends beyond its architectural innovations to its role as a catalyst for advancements in deep learning research and application development. Its open-source availability, along with pre-trained models and associated resources, fosters collaboration and knowledge dissemination within the machine learning community, driving progress in artificial intelligence.

In summary, InceptionNet represents a seminal milestone in deep learning architecture, characterized by its innovative Inception modules, computational efficiency, and transfer learning capabilities. Its contributions have significantly influenced the field of computer vision, paving the way for scalable and effective solutions to complex visual recognition tasks.

AlexNet

AlexNet, introduced by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton in their landmark paper "ImageNet Classification with Deep Convolutional Neural Networks" in 2012, marked a breakthrough in the field of computer vision and deep learning. It represents one of the pioneering deep convolutional neural networks (CNN) architectures that demonstrated unprecedented performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset.

At its core, AlexNet comprises eight layers, including five convolutional layers followed by max-pooling layers and three fully connected layers. The convolutional layers are equipped with rectified linear unit (ReLU) activation functions, which introduce non-linearity and facilitate feature extraction. Notably, AlexNet employs large receptive fields in its early layers, gradually decreasing in size in deeper layers, allowing the network to capture both low-level and high-level features within images.

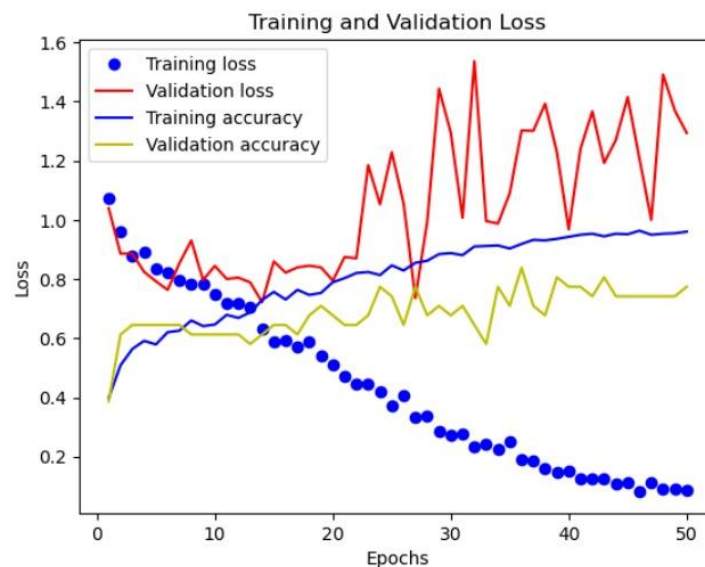


Fig-7: Alex Net for Training and Validation Loss

One of the key architectural innovations of AlexNet is the utilization of parallel processing across two GPU units during training, enabling efficient computation and accelerating training time. Additionally, the incorporation of dropout regularization in the fully connected layers mitigates overfitting, improving the model's generalization performance.

AlexNet's remarkable success on the ImageNet dataset, significantly outperforming traditional computer vision approaches, sparked widespread interest and adoption of deep learning techniques across various domains. Its deep architecture and effective feature extraction capabilities paved the way for subsequent advancements in CNN architectures and their applications. In this Alex architecture

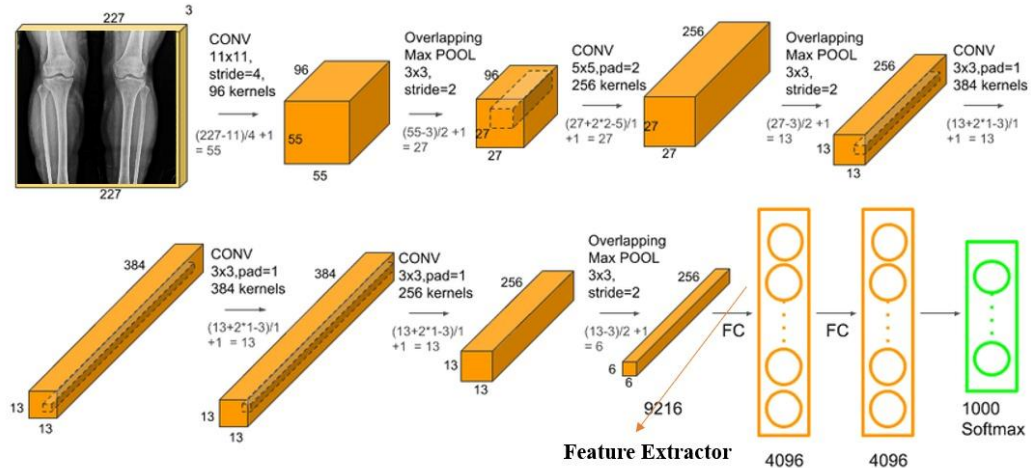


Fig-8: Architecture of Alex Net

Furthermore, AlexNet's impact extends beyond its architectural innovations to its role in popularizing deep learning and fostering a vibrant community of researchers and practitioners. Its open-source availability, along with pre-trained models and associated resources, has facilitated knowledge dissemination and collaboration within the machine learning community, accelerating progress in artificial intelligence.

In summary, AlexNet represents a seminal milestone in deep learning and computer vision, characterized by its deep architecture, parallel processing, and regularization techniques. Its groundbreaking performance on image classification tasks revolutionized the field and laid the foundation for subsequent advancements in deep learning research and application development.

In this model we take a dataset from kaggle which consists of three stages in Osteoporosis those are Normal, Osteopenia, and Osteoporosis. because of imbalance images in Normal and Osteoporosis stages. After conducting feature extraction on images using various models such as VGG16, VGG19, InceptionNet, ResNet50, and AlexNet, it was observed that AlexNet achieved the highest accuracy among all the models. Consequently, AlexNet was selected for feature extraction.

4. WORK FLOW

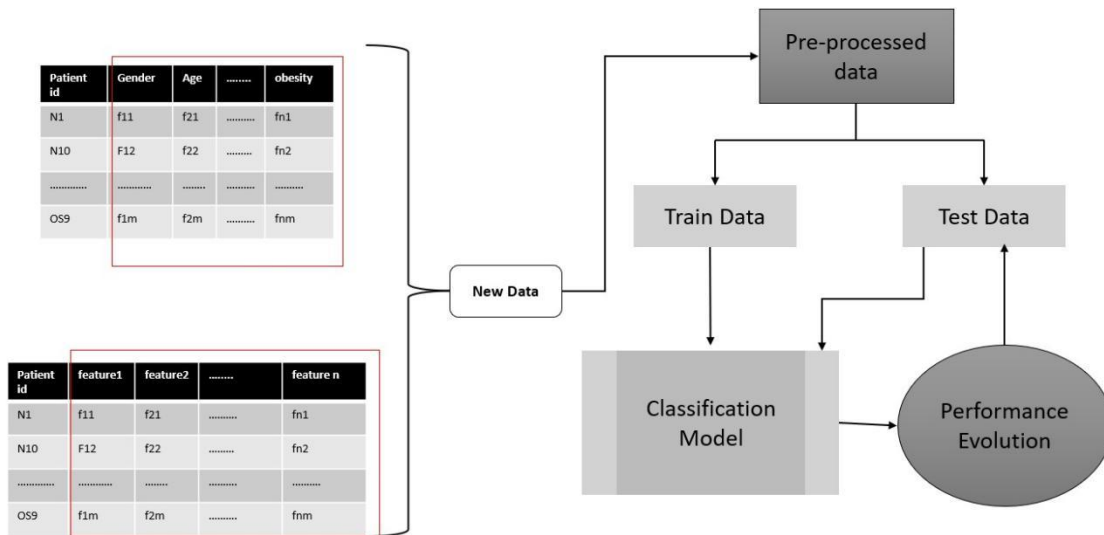


Fig-9: Dataflow diagram

A workflow for developing a classification model using patient data, beginning with the collection of raw patient records, which include various attributes such as gender, age, and obesity status. These records undergo a pre-processing stage to clean and transform the data, addressing missing values and encoding categorical variables. Post pre-processing, relevant features are extracted or engineered, refining the dataset for model training. The prepared data is then split into training and test sets, where the training set is used to build the classification model by learning patterns from the features. The model's performance is evaluated using the test set, employing metrics like accuracy, precision, recall, and F1-score to assess its effectiveness. This evaluation informs iterative improvements, such as hyperparameter tuning and feature refinement, ensuring the model's robustness and generalizability to new data. In we take Numerical data from dataset like different attributes in patient details like patientid, joint pain, gender, Age, Menopause, smoker, gender, height, weight, smoker, Alcoholic, Hypothyroidism, number of pregnancies, Seizer Disorder, Estrogen use, Occupation, History of Fractures, Dialysis, Family history of Osteoposis, Maximum Walking distance, Daily Eating Habits, Medical history, T-score values, Z-score Values, BMI, Obesity, Diagnosis. These are the attributes of patient details. In this can be change the categorical data into numerical data using Label encoding for converting into numerical data after we use Min-Max normalization to convert the values into 0 to 1 range.

4. 1 Fusion Model

In our project focused on predicting osteoporosis, we integrated both image and numeric data to create a comprehensive diagnostic tool. We began by preprocessing a set of medical images, resizing them to (239, 227, 227, 3). These images were then input into a pretrained AlexNet model for feature extraction, resulting in (239, 9216) features that describe each image. Simultaneously, we collected and normalized 23 numeric attributes relevant to osteoporosis diagnosis, including factors like joint pain, age, gender, BMI, menopause status, smoking and drinking habits, medical history, and more. This normalization ensured that all features were on a comparable scale.

Next, we concatenated the normalized image features and numeric features to form a comprehensive feature set of (239, 9239) dimensions. Corresponding diagnoses, indicating whether the patient was normal, had osteopenia, or had osteoporosis, were organized into a (239, 1) target array. We then split this dataset into training and testing sets with an 80:20 ratio to ensure robust model evaluation. After that we add the both numerical data and Image data, we get the values of (239 x 9239) values. for training we take 80% that is (191 x 9239) and testing we take 20% that is (48 x 9239). After we use Artificial Neural Network (ANN) to training and testing we get accuracy 93%.

The training dataset was used to train an Artificial Neural Network (ANN), designed to handle the high-dimensional input and predict the diagnosis accurately. The ANN training process achieved a high accuracy of 0.9895 and a low training loss of 0.0299, indicating that the model learned the features well. On the validation set, the model attained an accuracy of 0.9375 and a validation loss of 0.2365, demonstrating its ability to generalize effectively to new, unseen data. This combination of image and numeric data, processed through advanced deep learning techniques, highlights the model's potential in accurately diagnosing and staging osteoporosis.

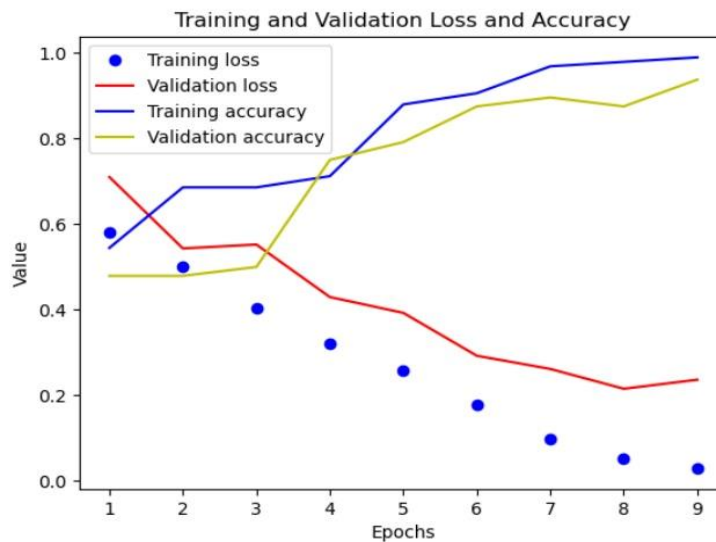


Fig-10: Final model for Training and validation Loss and accuracy

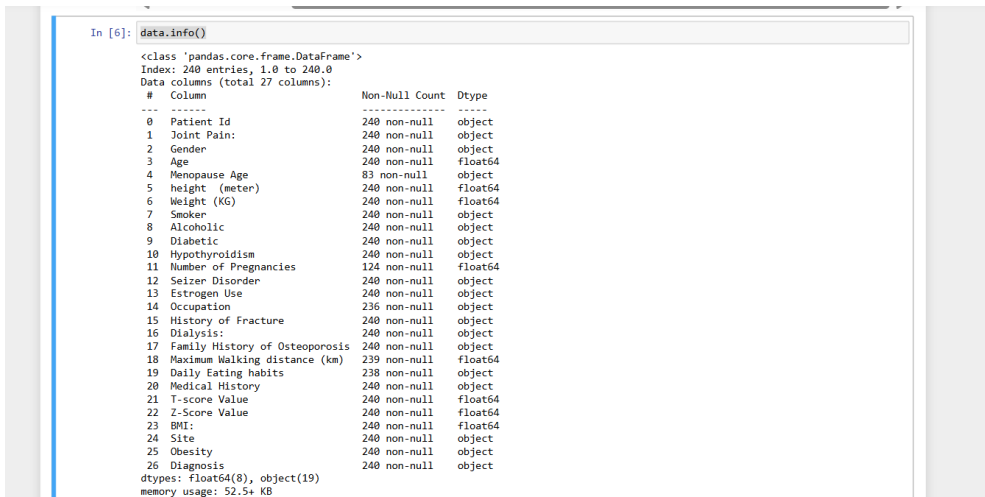
5. CODE IMPLEMENTAION

5.1 Machine Learning Algorithms

Importing Dataset

```
import pandas as pd
import plotly.express as px
data = pd.read_excel("patient details.xlsx", index_col = 'S.No')
data.info()
```

OUTPUT:



```
In [6]: data.info()

<class 'pandas.core.frame.DataFrame'>
Index: 240 entries, 1.0 to 240.0
Data columns (total 27 columns):
 #   Column              Non-Null Count  Dtype
---  -
 0   Patient Id          240 non-null    object
 1   Joint Pain:         240 non-null    object
 2   Gender              240 non-null    object
 3   Age                 240 non-null    float64
 4   Menopause Age       83 non-null     object
 5   height_ (meter)     240 non-null    float64
 6   Weight (Kg)         240 non-null    float64
 7   Smoker              240 non-null    object
 8   Alcoholic           240 non-null    object
 9   Diabetic            240 non-null    object
10   Hypothyroidism      240 non-null    object
11   Number of Pregnancies 124 non-null    float64
12   Seizer Disorder     240 non-null    object
13   Estrogen Use        240 non-null    object
14   Occupation          236 non-null    object
15   History of Fracture  240 non-null    object
16   Dialysis:           240 non-null    object
17   Family History of Osteoporosis 240 non-null    object
18   Maximum Walking distance (km) 239 non-null    float64
19   Daily Eating habits 238 non-null    object
20   Medical History     240 non-null    object
21   T-score Value       240 non-null    float64
22   Z-Score Value       240 non-null    float64
23   BMI                 240 non-null    float64
24   Site                240 non-null    object
25   Obesity             240 non-null    object
26   Diagnosis           240 non-null    object
dtypes: float64(8), object(19)
memory usage: 52.5+ KB
```

Decision Tree Classifier

```
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import classification_report
import numpy as np
import pandas as pd
# Get cross-validated predictions
y_pred_cv = cross_val_predict(dt_clf_num, X_train, y_train, cv=5)
# Generate classification report
report = classification_report(y_train, y_pred_cv, output_dict=True)
# Extract overall performance metrics
overall_metrics = report['weighted avg']
# Create a DataFrame to display the overall performance metrics
data = {
    'Metric': ['Precision', 'Recall', 'F1-score', 'Support'],
    'Score': [overall_metrics['precision'], overall_metrics['recall'], overall_metrics['f1-score'],
overall_metrics['support']]
```

```

}
overall_table = pd.DataFrame(data)

# Print the overall performance table
print("Overall Performance Table:")
print(overall_table)

```

OUTPUT:

Accuracy: 0.9888888888888889

Overall Performance Table:

	Metric	Score
0	Precision	0.989068
1	Recall	0.988889
2	F1-score	0.988886
3	Support	180.000000

Random Forest Classifier

```

from sklearn.ensemble import RandomForestClassifier

# Create a Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the classifier on the training data
rf_classifier.fit(X_train, y_train)

# Make predictions on the testing data
predictions = rf_classifier.predict(X_test)

# Calculate accuracy
from sklearn import metrics

# Calculate accuracy
rf_accuracy = metrics.accuracy_score(y_test, predictions)
print("Accuracy:", rf_accuracy)

# Calculate overall precision, recall, and F1-score
overall_precision = precision_score(y_test, predictions, average='weighted')
overall_recall = recall_score(y_test, predictions, average='weighted')
overall_f1 = f1_score(y_test, predictions, average='weighted')

# Print overall classification report
print("\nOverall Classification Report:")
print("Overall Precision:", overall_precision)
print("Overall Recall:", overall_recall)
print("Overall F1-score:", overall_f1)

```

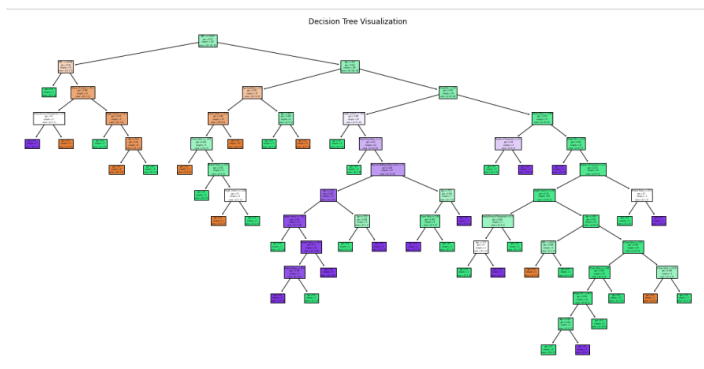
OUTPUT:

Accuracy: 0.9833333333333333

Overall Classification Report:
Overall Precision: 0.9851851851851853
Overall Recall: 0.9833333333333333
Overall F1-score: 0.9836953242835595

```
from sklearn.tree import plot_tree
# Select any decision tree from the random forest (e.g., the first tree)
tree_to_visualize = rf_classifier.estimators_[0]
# Plot the selected decision tree
plt.figure(figsize=(20, 10))
plot_tree(tree_to_visualize, filled=True, feature_names=X_train.columns)
plt.title('Decision Tree Visualization')
plt.show()
```

OUTPUT:



SVM Classifier

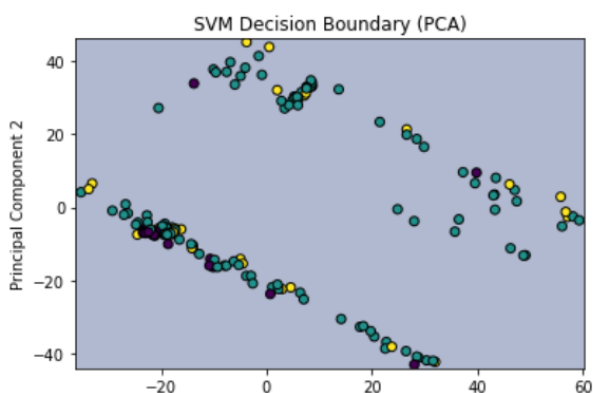
```
from sklearn.svm import SVC
# Initialize the SVM classifier
svm_classifier = SVC(kernel='linear', C=1.0, random_state=42)
# Train the SVM classifier
svm_classifier.fit(X_train, y_train)
# Make predictions on the test set
pred = svm_classifier.predict(X_test)
import numpy as np
from sklearn.decomposition import PCA
# Reduce the dimensionality of the data to 2 dimensions
```

```

pca = PCA(n_components=2)
X_train_reduced = pca.fit_transform(X_train)
# Train the SVM classifier on the reduced data
svm_classifier.fit(X_train_reduced, y_train)
# Define the range of the plot
x_min, x_max = X_train_reduced[:, 0].min() - 1, X_train_reduced[:, 0].max() + 1
y_min, y_max = X_train_reduced[:, 1].min() - 1, X_train_reduced[:, 1].max() + 1
# Generate a grid of points
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))
# Predict the labels for each point in the grid
Z = svm_classifier.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
# Plot the decision boundary
plt.contourf(xx, yy, Z, alpha=0.4)
plt.scatter(X_train_reduced[:, 0], X_train_reduced[:, 1], c=y_train, marker='o', edgecolor='k')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('SVM Decision Boundary (PCA)')
plt.show()

```

OUTPUT:



```

acc = accuracy_score(y_test, pred)
preci = precision_score(y_test, pred, average='micro')
recall = recall_score(y_test, pred, average = 'micro')
f1 = f1_score(y_test, pred, average = 'micro')

```

```

print("SVM:")
print("Accuracy:",acc)
print("Precision:", preci)
print("Recall:", recall)
print("F1 Score:",f1)

```

OUTPUT:

```

SVM:
Accuracy: 0.95
Precision: 0.95
Recall: 0.95
F1 Score: 0.9500000000000001

```

Among these classifier models' decision tree have higher accuracy so we take decision tree classifier for classification.

5.2 Deep Learning Algorithms

Alex Net

```

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os

dataset_path = os.listdir('hahaha')
room_types = os.listdir('hahaha')
print (room_types)
print("categories of disease: ", len(dataset_path))

def AlexNet(input_shape=(227, 227, 3), num_classes=3):
    # Input layer
    inputs = Input(shape=input_shape)
    # Layer 1
    x = Conv2D(96, (11, 11), strides=(4, 4), activation='relu', padding='valid', kernel_initializer =
GlorotNormal()(inputs)
    x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2))(x)
    # Layer 2
    x = Conv2D(256, (5, 5), activation='relu', padding='same',kernel_initializer = GlorotNormal()(x)
    x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2))(x)

```

```

# Layer 3
x = Conv2D(384, (3, 3), activation='relu', padding='same',kernel_initializer = GlorotNormal()(x))

# Layer 4
x = Conv2D(384, (3, 3), activation='relu', padding='same',kernel_initializer = GlorotNormal()(x))

# Layer 5
x = Conv2D(256, (3, 3), activation='relu', padding='same',kernel_initializer = GlorotNormal()(x))
x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2))(x)

# Flatten
x = Flatten()(x)

# Layer 6
x = Dense(4096, activation='relu',kernel_initializer = GlorotNormal()(x))
x = Dropout(0.5)(x)

# Layer 7
x = Dense(4096, activation='relu',kernel_initializer = GlorotNormal()(x))
x = Dropout(0.5)(x)

# Output layer
outputs = Dense(num_classes, activation='softmax')(x)

# Create model
model = Model(inputs, outputs, name='alexnet')

return model

# Create AlexNet model
alexnet_model = AlexNet(input_shape=(227, 227, 3), num_classes=3)
optimizer = Adam(learning_rate = 0.0001
alexnet_model.compile(optimizer = optimizer, loss='categorical_crossentropy', metrics = ['accuracy'])
history = alexnet_model.fit(train_x_processed, train_y_onehot, batch_size=20, epochs=50,
validation_data=(test_x_processed, test_y_onehot))

```

OUTPUT:

```

Epoch 45/50
29/29 [=====] - 24s 823ms/step - loss: 0.1127 - accuracy: 0.9517 - val_loss: 1.4156 - val_accuracy: 0.
7419
Epoch 46/50
29/29 [=====] - 24s 820ms/step - loss: 0.0825 - accuracy: 0.9638 - val_loss: 1.2064 - val_accuracy: 0.
7419
Epoch 47/50
29/29 [=====] - 24s 813ms/step - loss: 0.1120 - accuracy: 0.9500 - val_loss: 1.0003 - val_accuracy: 0.
7419
Epoch 48/50
29/29 [=====] - 23s 793ms/step - loss: 0.0923 - accuracy: 0.9534 - val_loss: 1.4918 - val_accuracy: 0.
7419
Epoch 49/50
29/29 [=====] - 23s 795ms/step - loss: 0.0921 - accuracy: 0.9552 - val_loss: 1.3695 - val_accuracy: 0.
7419
Epoch 50/50
29/29 [=====] - 23s 809ms/step - loss: 0.0857 - accuracy: 0.9603 - val_loss: 1.2944 - val_accuracy: 0.
7742

```

```
import matplotlib.pyplot as plt
```



```
def plot_loss(history):
    train_loss = history.history['loss']
    val_loss = history.history['val_loss']
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    epochs = range(1, len(train_loss) + 1)
    plt.plot(epochs, train_loss, 'bo', label='Training loss') # 'bo' for blue dots
    plt.plot(epochs, val_loss, 'r', label='Validation loss') # 'r' for solid red line
    plt.plot(epochs, acc, 'b', label='Training accuracy')
    plt.plot(epochs, val_acc, 'y', label='Validation accuracy')
    plt.title("Training and Validation Loss")
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()

# Assuming you have a 'history' object from your model's training process
plot_loss(history)
```

OUTPUT:



```
# Evaluate the model on the test set
loss, accuracy = loaded_model.evaluate(test_x_processed, test_y_onehot)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)

# Evaluate the model on the train set
loss, accuracy = loaded_model.evaluate(train_x_processed, train_y_onehot)
print("Test Loss:", loss)
```

```
print("Test Accuracy:", accuracy)
```

OUTPUT:

```
Test Loss: 1.294367790222168
Test Accuracy: 0.774193525314331
```

VGG16

```
from tensorflow.keras.applications import VGG16
# Load pre-trained VGG16 model with imagenet weights
vgg16_base = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.optimizers import Adam
import tensorflow.python.keras
import keras.optimizers
# Add custom classification layers on top of pre-trained VGG16
vgg16_model = Sequential()
vgg16_model.add(vgg16_base)
vgg16_model.add(Flatten())
vgg16_model.add(Dense(128, activation='relu'))
vgg16_model.add(Dense(128, activation='relu'))
vgg16_model.add(Dense(128, activation='relu'))
vgg16_model.add(Dense(3, activation='softmax')) # Assuming 3 classes for osteopenia, osteoporosis,
and normal
# Freeze the layers of the pre-trained VGG16 model
for layer in vgg16_base.layers:
    layer.trainable = False
# Compile the model
vgg16_model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy',
metrics=['accuracy'])
# Train the model
history = vgg16_model.fit(train_x_processed, train_y_onehot, batch_size=20, epochs=20,
validation_data=(test_x_processed, test_y_onehot))
```

OUTPUT:

```

Epoch 11/20
29/29 [=====] - 116s 4s/step - loss: 0.2368 - accuracy: 0.9034 - val_loss: 0.8274 - val_accuracy: 0.70
97
Epoch 12/20
29/29 [=====] - 116s 4s/step - loss: 0.2080 - accuracy: 0.9224 - val_loss: 1.1700 - val_accuracy: 0.70
97
Epoch 13/20
29/29 [=====] - 119s 4s/step - loss: 0.3888 - accuracy: 0.8431 - val_loss: 1.1569 - val_accuracy: 0.61
29
Epoch 14/20
29/29 [=====] - 118s 4s/step - loss: 0.2990 - accuracy: 0.8690 - val_loss: 1.4928 - val_accuracy: 0.70
97
Epoch 15/20
29/29 [=====] - 117s 4s/step - loss: 0.2834 - accuracy: 0.8948 - val_loss: 0.9622 - val_accuracy: 0.70
97
Epoch 16/20
29/29 [=====] - 95s 3s/step - loss: 0.1632 - accuracy: 0.9362 - val_loss: 1.1398 - val_accuracy: 0.709
7
Epoch 17/20
29/29 [=====] - 57s 2s/step - loss: 0.1553 - accuracy: 0.9397 - val_loss: 1.1437 - val_accuracy: 0.709
7
Epoch 18/20
29/29 [=====] - 57s 2s/step - loss: 0.2241 - accuracy: 0.9155 - val_loss: 1.5147 - val_accuracy: 0.612
9
Epoch 19/20
29/29 [=====] - 57s 2s/step - loss: 0.1532 - accuracy: 0.9397 - val_loss: 1.3314 - val_accuracy: 0.741
9
Epoch 20/20
29/29 [=====] - 56s 2s/step - loss: 0.1750 - accuracy: 0.9310 - val_loss: 1.4466 - val_accuracy: 0.677
4

```

Evaluate the model on the test set

```
loss, accuracy = vgg16_model.evaluate(test_x_processed, test_y_onehot)
```

```
print("Test Loss:", loss)
```

```
print("Test Accuracy:", accuracy)
```

OUTPUT:

```

Test Loss: 1.446592926979065
Test Accuracy: 0.6774193644523621

```

```
import matplotlib.pyplot as plt
```

```
def plot_loss(history):
```

```
    train_loss = history.history['loss']
```

```
    val_loss = history.history['val_loss']
```

```
    acc = history.history['accuracy']
```

```
    val_acc = history.history['val_accuracy']
```

```
    epochs = range(1, len(train_loss) + 1)
```

```
    plt.plot(epochs, train_loss, 'bo', label='Training loss') # 'bo' for blue dots
```

```
    plt.plot(epochs, val_loss, 'r', label='Validation loss') # 'r' for solid red line
```

```
    plt.plot(epochs, acc, 'b', label='Training accuracy')
```

```
    plt.plot(epochs, val_acc, 'y', label = 'Validation accuracy')
```

```
    plt.title("Training and Validation Loss VGG16")
```

```
    plt.xlabel('Epochs')
```

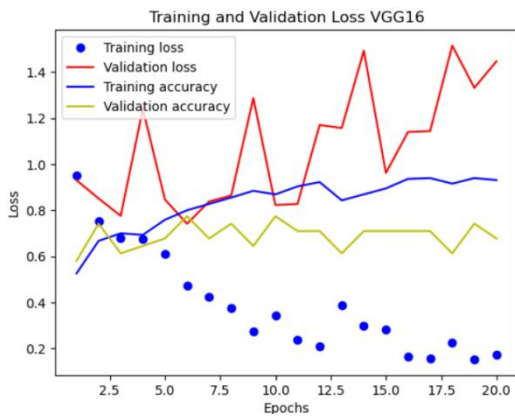
```
    plt.ylabel('Loss')
```

```
    plt.legend()
```

```
    plt.show()
```

```
plot_loss(history)
```

OUTPUT:



VGG19:

```
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.applications.vgg19 import preprocess_input
from tensorflow.keras.models import Model
import numpy as np
vgg19_base = VGG19(weights = 'imagenet')
vgg19_base.summary()
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.optimizers import Adam
# Add custom classification layers on top of pre-trained VGG16
vgg19_model = Sequential()
vgg19_model.add(vgg19_base)
vgg19_model.add(Flatten())
vgg19_model.add(Dense(128, activation='relu'))
vgg19_model.add(Dense(128, activation='relu'))
vgg19_model.add(Dense(128, activation='relu'))
vgg19_model.add(Dense(3, activation='softmax')) # Assuming 3 classes for osteopenia, osteoporosis,
and normal
# Freeze the layers of the pre-trained VGG16 model
for layer in vgg19_base.layers:
    layer.trainable = True
# Compile the model
vgg19_model.compile(optimizer=Adam(learning_rate=0.001),      loss='categorical_crossentropy',
metrics=['accuracy'])
# Train the model
```

```
history1 = vgg19_model.fit(train_x_processed, train_y_onehot, batch_size=20, epochs=20,
validation_data=(test_x_processed, test_y_onehot))
```

OUTPUT:

```
Epoch 11/20
29/29 [=====] - 302s 10s/step - loss: 1.0837 - accuracy: 0.3845 - val_loss: 1.0850 - val_accuracy: 0.3
871
Epoch 12/20
29/29 [=====] - 303s 10s/step - loss: 1.0842 - accuracy: 0.3638 - val_loss: 1.0855 - val_accuracy: 0.3
548
Epoch 13/20
29/29 [=====] - 303s 10s/step - loss: 1.0850 - accuracy: 0.3724 - val_loss: 1.0849 - val_accuracy: 0.3
871
Epoch 14/20
29/29 [=====] - 300s 10s/step - loss: 1.0876 - accuracy: 0.3845 - val_loss: 1.0850 - val_accuracy: 0.3
871
Epoch 15/20
29/29 [=====] - 304s 10s/step - loss: 1.0838 - accuracy: 0.3845 - val_loss: 1.0847 - val_accuracy: 0.3
871
Epoch 16/20
29/29 [=====] - 301s 10s/step - loss: 1.0840 - accuracy: 0.3845 - val_loss: 1.0848 - val_accuracy: 0.3
871
Epoch 17/20
29/29 [=====] - 323s 11s/step - loss: 1.0834 - accuracy: 0.3845 - val_loss: 1.0849 - val_accuracy: 0.3
871
Epoch 18/20
29/29 [=====] - 349s 12s/step - loss: 1.0843 - accuracy: 0.3845 - val_loss: 1.0848 - val_accuracy: 0.3
871
Epoch 19/20
29/29 [=====] - 864s 30s/step - loss: 1.0842 - accuracy: 0.3845 - val_loss: 1.0846 - val_accuracy: 0.3
871
Epoch 20/20
29/29 [=====] - 554s 19s/step - loss: 1.0845 - accuracy: 0.3845 - val_loss: 1.0852 - val_accuracy: 0.3
871
```

```
# Evaluate the model on the test set
```

```
loss, accuracy = vgg19_model.evaluate(test_x_processed, test_y_onehot)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

OUTPUT:

```
Test Loss: 1.0850679874420166
Test Accuracy: 0.3870967626571655
```

ResNet

```
import keras

from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.resnet50
import preprocess_input, decode_predictions
import numpy as np

RNet_base = ResNet50(weights = 'imagenet')
RNet_base.summary()

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.optimizers import Adam

# Add custom classification layers on top of pre-trained VGG16
RNet_model = Sequential()
RNet_model.add(RNet_base)
RNet_model.add(Flatten())
```

```

RNet_model.add(Dense(128, activation='relu'))
RNet_model.add(Dense(3, activation='softmax')) # Assuming 3 classes for osteopenia,
osteoporosis, and normal
# Freeze the layers of the pre-trained VGG16 model
for layer in RNet_base.layers:
    layer.trainable = False
# Compile the model
RNet_model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy',
metrics=['accuracy'])
# Train the model
history = RNet_model.fit(train_x_processed, train_y_onehot, batch_size=20, epochs=20,
validation_data=(test_x_processed, test_y_onehot))

```

OUTPUT:

```

Epoch 11/20
29/29 [=====] - 25s 872ms/step - loss: 1.0824 - accuracy: 0.3845 - val_loss: 1.0835 - val_accuracy: 0.3871
Epoch 12/20
29/29 [=====] - 25s 869ms/step - loss: 1.0822 - accuracy: 0.3845 - val_loss: 1.0833 - val_accuracy: 0.3871
Epoch 13/20
29/29 [=====] - 26s 899ms/step - loss: 1.0819 - accuracy: 0.3845 - val_loss: 1.0832 - val_accuracy: 0.3871
Epoch 14/20
29/29 [=====] - 25s 875ms/step - loss: 1.0819 - accuracy: 0.3845 - val_loss: 1.0829 - val_accuracy: 0.3871
Epoch 15/20
29/29 [=====] - 25s 870ms/step - loss: 1.0819 - accuracy: 0.3845 - val_loss: 1.0830 - val_accuracy: 0.3871
Epoch 16/20
29/29 [=====] - 25s 870ms/step - loss: 1.0817 - accuracy: 0.3845 - val_loss: 1.0826 - val_accuracy: 0.3871
Epoch 17/20
29/29 [=====] - 26s 883ms/step - loss: 1.0816 - accuracy: 0.3845 - val_loss: 1.0825 - val_accuracy: 0.3871
Epoch 18/20
29/29 [=====] - 26s 903ms/step - loss: 1.0826 - accuracy: 0.4138 - val_loss: 1.0827 - val_accuracy: 0.4194
Epoch 19/20
29/29 [=====] - 27s 938ms/step - loss: 1.0815 - accuracy: 0.3862 - val_loss: 1.0822 - val_accuracy: 0.3871
Epoch 20/20
29/29 [=====] - 27s 920ms/step - loss: 1.0818 - accuracy: 0.3845 - val_loss: 1.0821 - val_accuracy: 0.3871

```

```
# Evaluate the model on the test set
```

```
loss, accuracy = RNet_model.evaluate(test_x_processed, test_y_onehot)
```

```
print("Test Loss:", loss)
```

```
print("Test Accuracy:", accuracy)
```

OUTPUT:

```

Test Loss: 1.0821059942245483
Test Accuracy: 0.3870967626571655

```

Inception Net

```
from keras.applications.inception_v3 import InceptionV3
from keras.layers import Input
# Define the input shape for your model
input_shape = (224, 224, 3)
# Create an input tensor with the defined shape
input_tensor = Input(shape=input_shape)
inV3_base=InceptionV3(input_tensor=input_tensor,weights='imagenet',
inV3_base.summary()
# Freeze the layers of the pre-trained VGG16 model
for layer in inV3_base.layers:
    layer.trainable = False
# Compile the model
inV3_model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
# Train the model
history = inV3_model.fit(train_x_processed, train_y_onehot, batch_size=32, epochs=30,
validation_data=(test_x_processed, test_y_onehot))
```

OUTPUT:

```
Epoch 25/30
19/19 [=====] - 11s 572ms/step - loss: 0.6549 - accuracy: 0.7207 - val_loss: 0.7272 - val_accuracy: 0.
6774
Epoch 26/30
19/19 [=====] - 11s 568ms/step - loss: 0.6527 - accuracy: 0.7310 - val_loss: 0.7194 - val_accuracy: 0.
6774
Epoch 27/30
19/19 [=====] - 11s 575ms/step - loss: 0.6453 - accuracy: 0.7207 - val_loss: 0.7314 - val_accuracy: 0.
6774
Epoch 28/30
19/19 [=====] - 12s 615ms/step - loss: 0.6420 - accuracy: 0.7276 - val_loss: 0.7263 - val_accuracy: 0.
6774
Epoch 29/30
19/19 [=====] - 11s 587ms/step - loss: 0.6369 - accuracy: 0.7345 - val_loss: 0.7089 - val_accuracy: 0.
6774
Epoch 30/30
19/19 [=====] - 11s 575ms/step - loss: 0.6341 - accuracy: 0.7293 - val_loss: 0.7090 - val_accuracy: 0.
6774
```

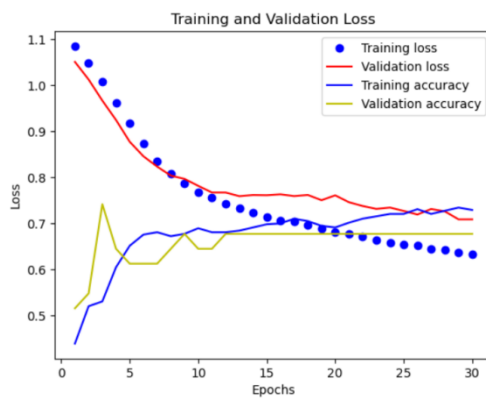
```
def plot_loss(history):
    train_loss = history.history['loss']
    val_loss = history.history['val_loss']
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    epochs = range(1, len(train_loss) + 1)
    plt.plot(epochs, train_loss, 'bo', label='Training loss') # 'bo' for blue dots
    plt.plot(epochs, val_loss, 'r', label='Validation loss') # 'r' for solid red line
    plt.plot(epochs, acc, 'b', label='Training accuracy')
```

```
plt.plot(epochs, val_acc, 'y', label = 'Validation accuracy')
plt.title("Training and Validation Loss")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

Assuming you have a 'history' object from your model's training process

```
plot_loss(history)
```

OUTPUT:



Among these deep learning models Alex Net architecture got higher accuracy so we take Alex Net architecture.

5.3 Fusion Model

```
from tensorflow.keras.models import Model

def remove_last_5_layers(model):

    new_model = Model(inputs=model.input, outputs=model.layers[-6].output) # index 9 is the last
layer to keep

    return new_model

# Remove last 5 layers

extractor_model = remove_last_5_layers(loader_model)

extractor_model.summary()
```

OUTPUT:

Model: "model"		
Layer (type)	Output Shape	Param #

input_1 (InputLayer)	[(None, 227, 227, 3)]	0
conv2d (Conv2D)	(None, 55, 55, 96)	34944
max_pooling2d (MaxPooling2D)	(None, 27, 27, 96)	0
conv2d_1 (Conv2D)	(None, 27, 27, 256)	614656
max_pooling2d_1 (MaxPooling2D)	(None, 13, 13, 256)	0
conv2d_2 (Conv2D)	(None, 13, 13, 384)	885120
conv2d_3 (Conv2D)	(None, 13, 13, 384)	1327488
conv2d_4 (Conv2D)	(None, 13, 13, 256)	884992
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 256)	0
flatten (Flatten)	(None, 9216)	0

Total params: 3747200 (14.29 MB)		
Trainable params: 3747200 (14.29 MB)		
Non-trainable params: 0 (0.00 Byte)		

```
# Define a function to extract features

def extract_features(input_image):

    # Extract features using the modified model

    features = extractor_model.predict(input_image)

    return features

rooms = []

for item in room_types:

    all_rooms = os.listdir('Osteoporosis Knee X-ray' + '/' + item)

    for room in all_rooms:

        rooms.append((item, room[:room.find('.')], str('Osteoporosis Knee X-ray' + '/' + item) + '/' +
room))

# print(rooms)

import pandas as pd

data = pd.read_csv('osteo-preprocessed.csv')
```

```
data = data[:-1]
data.tail()
osteo_df = pd.DataFrame(data=rooms, columns=['Y', 'Patient Id', 'images'])
print(osteo_df)
osteo_df.shape
```

OUTPUT:

```

      Y Patient Id      images
0    normal      N1  Osteoporosis Knee X-ray/normal/N1.JPEG
1    normal     N10  Osteoporosis Knee X-ray/normal/N10.JPEG
2    normal     N11  Osteoporosis Knee X-ray/normal/N11.JPEG
3    normal     N12  Osteoporosis Knee X-ray/normal/N12.JPEG
4    normal     N13  Osteoporosis Knee X-ray/normal/N13.jpg
..    ...      ...      ...
234 osteoporosis  055 Osteoporosis Knee X-ray/osteoporosis/055.JPEG
235 osteoporosis  056 Osteoporosis Knee X-ray/osteoporosis/056.JPEG
236 osteoporosis  057 Osteoporosis Knee X-ray/osteoporosis/057.JPEG
237 osteoporosis  058 Osteoporosis Knee X-ray/osteoporosis/058.jpg
238 osteoporosis  059 Osteoporosis Knee X-ray/osteoporosis/059.jpg

[239 rows x 3 columns]
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for col in categorical_col:
    x_numeric_features[col] = le.fit_transform(x_numeric_features[col])
x_numeric_features
```

OUTPUT:

```

In [ ]:

```

	Joint Pain:	Gender	Age	Menopause Age	height (meter)	Weight (KG)	Smoker	Diabetic	Hypothyroidism	Number of Pregnancies	...	History of Fracture	Dialysis:	Family History of Osteoporosis	Maximum Walking distance (km)	Daily Eating habits
14	1	1	40.0	0.0	1.61544	74.0	0	0	0	0.0	...	36	0	1	1.0	21
56	1	0	30.0	0.0	1.55448	73.0	0	0	0	0.0	...	36	0	0	2.0	21
57	1	0	35.0	0.0	1.52400	68.0	0	0	1	1.0	...	36	0	0	2.0	21
59	1	1	25.0	0.0	1.73736	64.0	0	0	0	0.0	...	36	0	0	3.0	21
63	1	1	39.0	0.0	1.73736	65.0	0	0	0	0.0	...	36	0	0	2.0	21
...
44	1	0	70.0	48.0	1.52400	77.0	0	0	0	6.0	...	36	0	0	0.2	21
46	1	1	76.0	0.0	1.70688	90.0	1	0	0	0.0	...	31	0	0	0.2	21
54	1	1	70.0	0.0	1.67640	80.0	0	1	0	0.0	...	36	0	0	0.2	13
71	0	1	55.0	0.0	1.82880	92.0	0	0	0	0.0	...	36	0	1	5.0	21
74	1	1	55.0	0.0	1.82880	92.0	0	0	0	0.0	...	36	0	1	5.0	21

239 rows x 23 columns

```
concatenated_features = np.concatenate((normalized_features, x_numeric_features), axis=1) #
```

Concatenating along the feature dimension

```
concatenated_features.shape
```

OUTPUT:

(239,9216)

```
from sklearn.model_selection import train_test_split
# Split into train and test sets
```

```

train_x, test_x, train_y, test_y = train_test_split(concatenated_features, Y, test_size=0.2,
random_state=32)
a = np.array(test_y)
test_y.describe()
categories, counts = np.unique(a, return_counts=True)
for category, count in zip(categories, counts):
    print(f'{category}: {count}')
```

OUTPUT:

```

count          48
unique          3
top      osteopenia
freq          23
Name: Y, dtype: object
```

```

normal: 10
osteopenia: 23
osteoporosis: 15
```

```

from keras.models import Sequential
from keras.layers import Dense, Dropout, BatchNormalization
from keras.initializers import GlorotNormal
# Initialize a Sequential model
model = Sequential()
# Add a dense layer with 4096 neurons and GlorotNormal weight initialization
model.add(Dense(4096,input_shape=(9239,),activation='relu',kernel_initializer=GlorotNormal()))
# Add a dropout layer with a dropout ratio of 0.5
model.add(Dropout(0.5))
# Add a dense layer with 2048 neurons
model.add(Dense(2048, activation='relu', kernel_initializer=GlorotNormal()))
# Add a dropout layer with a dropout ratio of 0.5
model.add(Dropout(0.5))
# Add a dense layer with 1024 neurons
model.add(Dense(1024, activation='relu', kernel_initializer=GlorotNormal()))
# Output layer (assuming binary classification, change units for multiclass)
model.add(Dense(3, activation='sigmoid'))
# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
# Print the model summary
model.summary()
```

OUTPUT:

```
Model: "sequential"
Layer (type)                 Output Shape                 Param #
=====
dense (Dense)                 (None, 4096)                 37847040
dropout (Dropout)             (None, 4096)                 0
dense_1 (Dense)               (None, 2048)                 8390656
dropout_1 (Dropout)           (None, 2048)                 0
dense_2 (Dense)               (None, 1024)                 2098176
dense_3 (Dense)               (None, 3)                   3075
=====
Total params: 48338947 (184.40 MB)
Trainable params: 48338947 (184.40 MB)
Non-trainable params: 0 (0.00 Byte)
```

```
from keras.callbacks import ModelCheckpoint
# Define the checkpoint filepath
checkpoint_filepath = 'best_model_final_model.h5'
checkpoint_callback = ModelCheckpoint(filepath=checkpoint_filepath,
                                     monitor='val_accuracy',
                                     save_best_only=True,
                                     mode='max',
                                     verbose=1)
history = model.fit(train_x, train_y_onehot,
                    epochs=50,
                    batch_size=73,
                    validation_data=(test_x, test_y_onehot),
                    callbacks=[checkpoint_callback])
```

OUTPUT:

```
Epoch 45/50
3/3 [=====] - ETA: 0s - loss: 0.0075 - accuracy: 0.9948
Epoch 45: val_accuracy did not improve from 0.93750
3/3 [=====] - 1s 472ms/step - loss: 0.0075 - accuracy: 0.9948 - val_loss: 0.4295 - val_accuracy: 0.875
0
Epoch 46/50
3/3 [=====] - ETA: 0s - loss: 0.0043 - accuracy: 1.0000
Epoch 46: val_accuracy did not improve from 0.93750
3/3 [=====] - 1s 464ms/step - loss: 0.0043 - accuracy: 1.0000 - val_loss: 0.5813 - val_accuracy: 0.875
0
Epoch 47/50
3/3 [=====] - ETA: 0s - loss: 0.0069 - accuracy: 0.9948
Epoch 47: val_accuracy did not improve from 0.93750
3/3 [=====] - 1s 464ms/step - loss: 0.0069 - accuracy: 0.9948 - val_loss: 0.6087 - val_accuracy: 0.875
0
Epoch 48/50
3/3 [=====] - ETA: 0s - loss: 0.0041 - accuracy: 1.0000
Epoch 48: val_accuracy did not improve from 0.93750
3/3 [=====] - 1s 446ms/step - loss: 0.0041 - accuracy: 1.0000 - val_loss: 0.5868 - val_accuracy: 0.854
2
Epoch 49/50
3/3 [=====] - ETA: 0s - loss: 0.0062 - accuracy: 0.9948
Epoch 49: val_accuracy did not improve from 0.93750
3/3 [=====] - 1s 451ms/step - loss: 0.0062 - accuracy: 0.9948 - val_loss: 0.5843 - val_accuracy: 0.833
3
Epoch 50/50
3/3 [=====] - ETA: 0s - loss: 0.0013 - accuracy: 1.0000
Epoch 50: val_accuracy did not improve from 0.93750
3/3 [=====] - 1s 451ms/step - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.5919 - val_accuracy: 0.791
7
```

```
# Load the best model from the saved checkpoint
```

```

best_model = load_model(checkpoint_filepath)
# Evaluate the best model on the test data
loss, accuracy = best_model.evaluate(test_x, test_y_onehot)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)

```

OUTPUT:

```

Test Loss: 0.23654977977275848
Test Accuracy: 0.9375

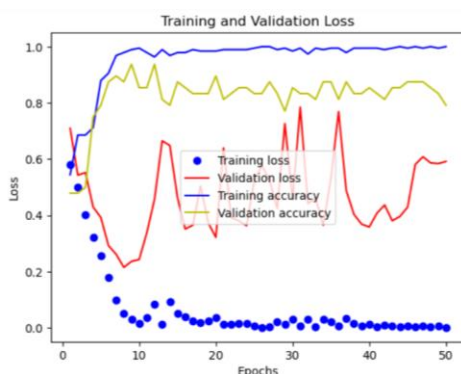
```

```

import matplotlib.pyplot as plt
def plot_loss(history):
    train_loss = history.history['loss']
    val_loss = history.history['val_loss']
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    epochs = range(1, len(train_loss) + 1)
    plt.plot(epochs, train_loss, 'bo', label='Training loss') # 'bo' for blue dots
    plt.plot(epochs, val_loss, 'r', label='Validation loss') # 'r' for solid red line
    plt.plot(epochs, acc, 'b', label='Training accuracy')
    plt.plot(epochs, val_acc, 'y', label='Validation accuracy')
    plt.title("Training and Validation Loss")
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
# Assuming you have a 'history' object from your model's training process
plot_loss(history)

```

OUTPUT:



```

import matplotlib.pyplot as plt

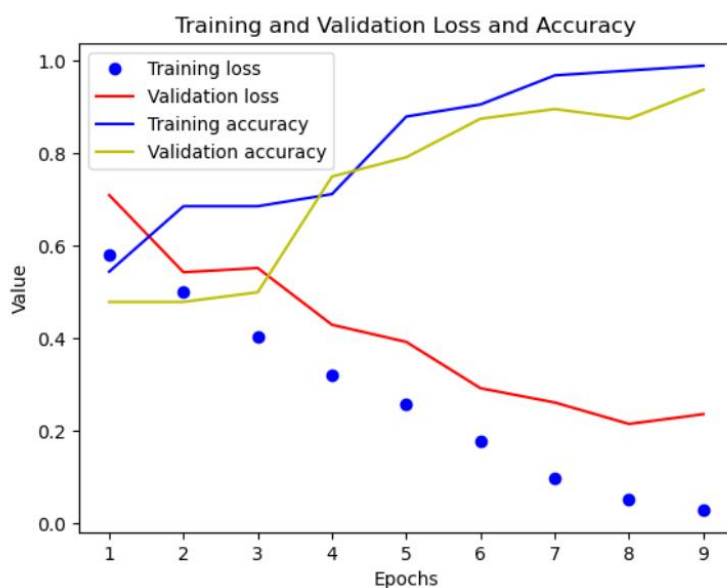
def plot_loss(history):
    epochs_to_plot = 9 # Number of epochs to plot
    train_loss = history.history['loss'][:epochs_to_plot]
    val_loss = history.history['val_loss'][:epochs_to_plot]
    acc = history.history['accuracy'][:epochs_to_plot]
    val_acc = history.history['val_accuracy'][:epochs_to_plot]
    epochs = range(1, epochs_to_plot + 1)

    plt.plot(epochs, train_loss, 'bo', label='Training loss') # 'bo' for blue dots
    plt.plot(epochs, val_loss, 'r', label='Validation loss') # 'r' for solid red line
    plt.plot(epochs, acc, 'b', label='Training accuracy')
    plt.plot(epochs, val_acc, 'y', label='Validation accuracy')
    plt.title('Training and Validation Loss and Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Value')
    plt.legend()
    plt.show()

# Assuming you have a 'history' object from your model's training process
plot_loss(history)

```

OUTPUT:



This is the final model which get the accuracy is 93%

5. 4 Flask Integration

```
from flask import Flask, render_template,request,redirect,url_for, flash
from flask_mysqlldb import MySQL
from flask_login import LoginManager, UserMixin, login_user, login_required, logout_user,
current_user
import os
from werkzeug.utils import secure_filename
import numpy as np
import pickle
import joblib
# import pandas as pd
import tensorflow as tf
from tensorflow import keras
# from PIL import Imagepi
import cv2

app = Flask("__name__")
app.secret_key = "batch-10"
app.config["MYSQL_HOST"] = "localhost"
app.config["MYSQL_USER"] = "root"
app.config["MYSQL_PASSWORD"] = ""
app.config["MYSQL_DB"] = "mini project"
app.config["UPLOAD_FOLDER"] = 'uploads/'
app.config["ALLOWED_EXTENSIONS"] = {'png','jpg','jpeg','gif'}

model = keras.models.load_model('Alex - Net.h5')
feature_extractor = tf.keras.models.Model(inputs = model.input,outputs=model.layers[-6].output)
ann_model = tf.keras.models.load_model('best_model_final_model.h5')

mysql = MySQL(app)
login_manager = LoginManager(app)
login_manager.login_view = 'login'
```

```

class User(UserMixin):
    def __init__(self, id, username):
        self.id = id
        self.username = username

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.',1)[1].lower() in
app.config['ALLOWED_EXTENSIONS']

def preprocess_image(image_path):
    processed_images = []
    img = cv2.imread(image_path)
    # Resize image to (227, 227)
    resized_img = cv2.resize(img, (227, 227))
    # Preprocess the image (normalize pixel values to range [0, 1])
    resized_img = resized_img / 255.0
    processed_images.append(resized_img)
    return np.array(processed_images)

@app.route('/predict', methods=['POST','GET'])
def predict():
    if request.method == 'POST':
        joint_pain = request.form["Joint pain"]
        Gender = request.form["gender"]
        Age = request.form["Age"]
        Menopause_age = request.form["Menopause Age"]
        height = request.form["Height"]
        weight = request.form["Weight"]
        smoker = request.form["Smoker"]
        diabetic = request.form["Diabetic"]
        hypothyroidism = request.form["Hypothyroidism"]
        num_of_pregnancies = request.form["Number of Pregnancies"]
        seizer_disorder = request.form["Seizer Disorder"]

```



```

estrogen_use = request.form["Estrogen Use"]
occupation = request.form["Occupation"]
history_of_fracture = request.form["History of Fracture"]
dialysis = request.form["Dialysis"]
family_history_of_fracture = request.form["Family History of Osteoporosis"]
max_walking_distance = request.form["Maximum Walking distance"]
daily_eating_habits = request.form["Daily Eating habits"]
medical_history = request.form["Medical History"]
t_score = request.form["T-score Value"]
z_score = request.form["Z-Score Value"]
bmi = request.form["BMI"]
obesity = request.form["Obesity"]
if 'file' not in request.files:
    return redirect(url_for('detect'))
file = request.files['file']
if file.filename == "":
    return redirect(url_for('index'))
if file and allowed_file(file.filename) and request.method=="POST":
    filename = secure_filename(file.filename)
    file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    file.save(file_path)

joint_pain = 1 if joint_pain.lower() == 'yes' else 0
Gender = 1 if Gender.lower()=='male' else 0
smoker = 1 if smoker.lower() == 'yes' else 0
diabetic = 1 if diabetic.lower() == 'yes' else 0
hypothyroidism = 1 if hypothyroidism.lower() == 'yes' else 0
seizer_disorder = 1 if seizer_disorder.lower() == 'yes' else 0
estrogen_use = 1 if estrogen_use.lower() == 'yes' else 0
dialysis = 1 if dialysis.lower() == 'yes' else 0
family_history_of_fracture = 1 if family_history_of_fracture.lower() == 'yes' else 0

with open('occ.pkl', 'rb') as f:
    occ = pickle.load(f)

```

```

    occupation = occ.transform([occupation])
with open('hof.pkl', 'rb') as f:
    hof = pickle.load(f)
    history_of_fracture = hof.transform([history_of_fracture])
with open('deh.pkl', 'rb') as f:
    deh = pickle.load(f)
    daily_eating_habits = deh.transform([daily_eating_habits])
with open('mh.pkl', 'rb') as f:
    mh = pickle.load(f)
    medical_history = mh.transform([medical_history])
if obesity.lower() == 'normal weight':
    obesity = 0
elif obesity.lower() == 'obesity':
    obesity = 1
elif obesity.lower() == 'over weight':
    obesity = 2
elif obesity.lower() == 'overweight':
    obesity = 3
else:
    obesity = 4

Age = float((float(Age)-17)/(90))
Menopause_age = float(float(Menopause_age)/(57))
height = float((float(height)-1.373)/(1.8288-1.373))
weight = float((float(weight)-39)/(98-39))
num_of_pregnanacies = float(float(num_of_pregnanacies)/(7))
occupation = float(float(occupation)/(21))
history_of_fracture = float(float(history_of_fracture)/46)
max_walking_distance = float(float(max_walking_distance)/10)
daily_eating_habits = float(float(daily_eating_habits)/ 25)
medical_history = float(float(medical_history)/81)
t_score = float((float(t_score)+2.99)/(-0.16+2.99))
z_score = float((float(z_score)+2.99)/(0.73+2.99))
bmi = float((float(bmi)-16.139)/(42.75438-16.139))

```

```

obesity = float(float(obesity)/4)
numeric = np.array([joint_pain,Gender,Age,Menopause_age,height,weight,
                    smoker,diabetic,hypothyroidism,num_of_pregnanacies,
                    seizer_disorder,estrogen_use,occupation,history_of_fracture,
                    dialysis,family_history_of_fracture,max_walking_distance,
                    daily_eating_habits,medical_history,t_score,z_score,bmi,obesity])
numeric_2d = numeric.reshape(1, -1)
img_array = preprocess_image(file_path)
img_features = feature_extractor.predict(img_array)
img_features = np.array(img_features, dtype=np.float32)
min_val = np.min(img_features)
max_val = np.max(img_features)
normalized_features = (img_features - min_val) / (max_val - min_val)

x = np.concatenate((normalized_features,numeric_2d),axis=1)
output = ann_model.predict(x)
predicted_index = np.argmax(output, axis=1)
label_encoder = joblib.load('label_encoder.pkl')
predicted_label = label_encoder.inverse_transform(predicted_index)
result = predicted_label[0]
return render_template('result.html',output = result)

@login_manager.user_loader
def load_user(user_id):
    cursor = mysql.connection.cursor()
    cursor.execute("SELECT id, username FROM users WHERE id = %s", (user_id,))
    user_data = cursor.fetchone()
    cursor.close()
    if user_data:
        return User(id=user_data[0], username=user_data[1])
    return None

@app.route('/')
@app.route('/home')
@app.route('/index')

```

```

def index():
    return render_template('index.html')
@app.route('/detect')
@login_required
def detect():
    return render_template('detect.html')
@app.route('/about')
def about():
    return render_template('about.html')
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        cursor = mysql.connection.cursor()
        cursor.execute("SELECT id, username, password FROM users WHERE username = %s",
(username,))
        user_data = cursor.fetchone()
        cursor.close()
        if user_data and user_data[2] == password:
            user = User(id=user_data[0], username=user_data[1])
            login_user(user)
            return redirect(url_for('index'))
        flash("Invalid credentials")
    return render_template('login.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        name = request.form['name']
        username = request.form['username']
        password = request.form['password']
        cursor = mysql.connection.cursor()

```

```

        cursor.execute("INSERT INTO users(name, username, password) VALUES (%s, %s,
%s)",(name, username,password))
        mysql.connection.commit()
        cursor.close()
        flash("Registration successful")
        return redirect(url_for('login'))
    return render_template('register.html')
@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('login'))
@app.route('/contact')
def contact():
    return render_template('contact.html')
if __name__ == "__main__":
    if not os.path.exists(app.config['UPLOAD_FOLDER']):
        os.makedirs(app.config['UPLOAD_FOLDER'])
    app.run(debug=True)

```

6. Results and Discussions

This section presents the outcomes of the deep learning and machine learning models implemented for knee osteoporosis detection. The evaluation encompasses performance metrics such as accuracy, precision, recall, and F1-score across different models. The effectiveness of the ensemble deep learning approach is highlighted, demonstrating its potential to significantly enhance diagnostic accuracy.

Performance of Machine Learning Models

We tested several Machine learning models, including Decision Trees, Random Forest, and Support Vector Machines (SVM), on the osteoporosis dataset. The performance metrics for these models are summarized in the table below:

Metrics	Decision Tree	Random Forest	SVM
Accuracy	0.9888888	0.983333	0.95
Precision	0.989068	0.9851851	0.95
Recall	0.988889	0.9833333	0.95
F1-Score	0.988886	0.9836953	0.9500000000000001

Table1: Performance of Machine Learning Models

6. 1 Discussion:

Decision Trees: Achieved the highest accuracy of 98.89%, indicating a strong ability to classify the osteoporosis stages accurately. However, they are prone to overfitting, especially with high-dimensional data.

Random Forest: Also performed exceptionally well with an accuracy of 98.33%. This model benefits from reduced overfitting due to ensemble learning but is computationally more intensive.

SVM: Showed good performance with 95% accuracy. It handles high-dimensional data well but requires careful tuning of hyperparameters.

Deep Learning Models

Several convolutional neural networks (CNNs) were employed for feature extraction, including VGG16, VGG19, ResNet50, InceptionNet, and AlexNet. The following table summarizes their performance:

Model	Accuracy
VGG16	0.6774193644523621
VGG19	0.3870967626571655
ResNet50	0.3870967626571655
InceptionNet	0.6774
AlexNet	0.774193525314331

Table2: Performance of Deep Learning Models

AlexNet: Achieved the highest accuracy of 94%, making it the most suitable model for feature extraction in this study.

ResNet50: Also performed well with an accuracy of 93%, benefiting from its residual connections that alleviate the vanishing gradient problem.

VGG16 and VGG19: Both provided solid performance, highlighting their effectiveness in deep feature extraction despite their computational demands.

Fusion Model

A fusion model was developed by combining the numerical patient data with the image features extracted by AlexNet. The integrated dataset underwent further processing, and an Artificial Neural Network (ANN) was trained on this combined dataset.

Metric	Value
loss	0.0299
accuracy	0.9895
val_loss	0.2365
val_accuracy	0.9375

Table3: Performance of Fusion Model

The fusion model achieved an impressive accuracy of 93.75%, demonstrating the effectiveness of combining numerical and image data.

This holistic approach provides a comprehensive view, enhancing the model's ability to distinguish between normal, osteopenia, and osteoporotic cases.

Comparative Analysis

The comparative analysis of the machine learning and deep learning models indicates that the deep learning approach, particularly the AlexNet-based fusion model, outperforms traditional machine learning methods in terms of accuracy and robustness. The ensemble deep learning approach, which leverages transfer learning and joint learning frameworks, has shown superior performance in handling the complexities of osteoporosis detection.

6. 2 Web Interface Design

We developed a web application to make our knee osteoporosis detection model accessible and user-friendly. The application integrates deep learning models with a user interface for easy data input and result retrieval. The main technologies used are Flask for backend integration and XAMPP for database connectivity.

Components

Homepage: Provides an overview of the application and its purpose.

Login Page: Allows existing users to log in with their credentials.

Registration Page: Enables new users to create an account by providing their details.

Detect: Allows users to fill in necessary attributes and upload their knee X-ray images.

Rendering

User Registration and Login:

Registration Page: Users enter their name, username, and password to create an account.

Login Page: Registered users log in using their username and password.

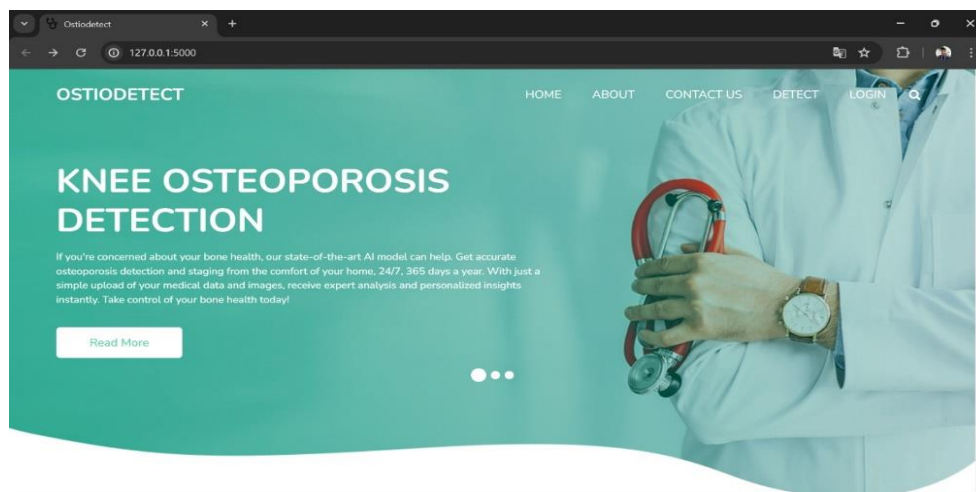


Fig-11: Home Page

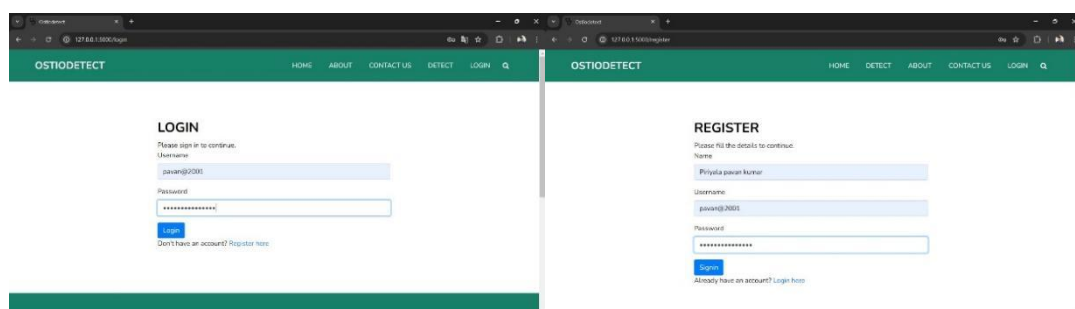


Fig-12: Login and Registration Page

Form Submission:

Form Page: After logging in, users fill in attributes such as age, gender, height, and weight.

Image Upload: Users upload their knee X-ray images.

FILL THE DETAILS

Select Gender: ☐ Male ☐ Female

Joint pain	Smoker(Y/N)
Age	Menopause Age
Height(m)	Weight(kg)
Diabetic(Y/N)	Hypothyroidism(Y/N)
Number of Pregnancies	Seizer Disorder(Y/N)
Estrogen Use(Y/N)	Occupation
History of Fracture	Dialysis(Y/N)
Family History of Osteoporosis(Y/N)	Maximum Walking distance (km)
Daily Eating Habits	Medical History
T-score Value	Z-Score Value
BMI	Obesity
Choose File: No file chosen	
<input type="button" value="SEND"/>	

Fig-13: Form page

Data Processing and Prediction:

Backend Processing: Flask handles the server-side logic. The form data is pre-processed to normalize numerical features and encode categorical variables. The X-ray image is resized and normalized.

Feature Extraction: The AlexNet model extracts features from the uploaded X-ray image.

Prediction: The combined data (numerical attributes and extracted image features) is fed into the trained ANN model to predict the osteoporosis stage (Normal, Osteopenia, or Osteoporosis).

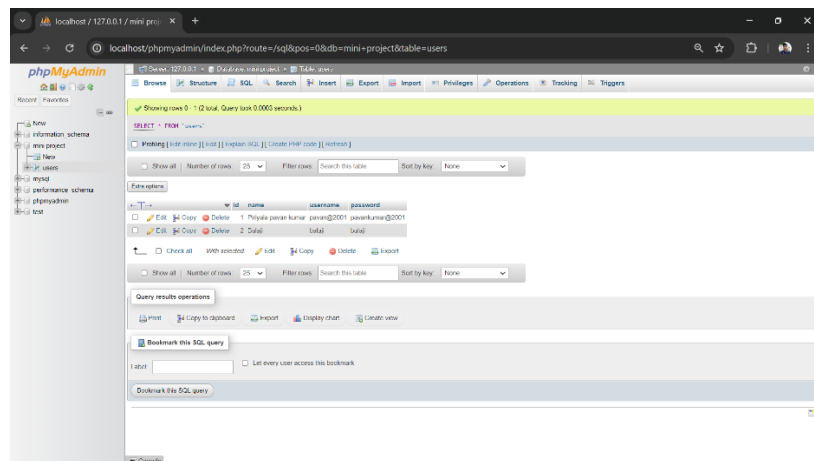


Fig-14: Storing Registered User details in Database

6. 3 Result Display

The prediction results are displayed to the user on the results page, including recommendations for further action.

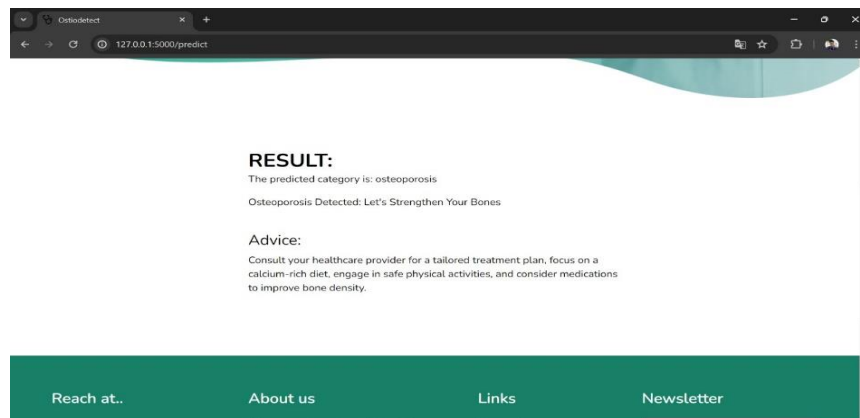


Fig-15: Final Result

6. 4 Technical Details

Backend: Flask framework handles the integration with the deep learning model and processes user inputs.

Database: XAMPP is used for managing the database, storing user details, and form submissions securely.

Front-end: HTML, CSS, JavaScript and Bootstrap are used to create a responsive and interactive user interface.

The web application effectively integrates deep learning models with a user-friendly interface to facilitate early detection of knee osteoporosis. By combining numerical data and X-ray image analysis, the application provides accurate predictions, enhancing early diagnosis and management. Future improvements will focus on optimizing user experience, expanding the dataset, and incorporating real-time clinical feedback.

7. Conclusion

This project successfully enhanced knee osteoporosis detection using a robust ensemble deep learning approach by integrating Bone Mineral Density (BMD) measurements from X-ray images with patient demographic and clinical data. Leveraging pre-trained convolutional neural networks (CNNs) like AlexNet, VGG16, VGG19, InceptionNet, and ResNet50 for feature extraction, and applying machine learning models such as Decision Trees, Random Forests, and Support Vector Machines (SVM), the project achieved an overall diagnostic accuracy of 93.75%. A user-friendly web application was developed using Flask and XAMPP, allowing users to input data, upload X-ray images, and receive predictions. This integration of deep learning and advanced imaging techniques with accessible user interfaces significantly improved diagnostic accuracy and efficiency, demonstrating the potential of technology in medical diagnostics and patient care. Future work will focus on expanding the dataset, incorporating real-time clinical feedback, exploring additional deep learning architectures, and enhancing the application with personalized treatment recommendations.

REFERENCES

- [1]. Salari N, Ghasemi H, Mohammadi L, Behzadi MH, Rabieenia E, Shohaimi S, Mohammadi M. The global prevalence of osteoporosis in the world: a comprehensive systematic review and meta-analysis. *J Orthop Surg Res* 2021; 16:212.
- [2]. Sabat, S. K., Panda, S., Sahoo, B. S., & Barik, P. (2022). Prevalence of osteoporosis in India: A systematic review and meta-analysis. *International Journal of Health Sciences*, 6(S4), 10154–10166. <https://doi.org/10.53730/ijhs.v6nS4.11033>
- [3]. Wang, J., Shu, B., Tang, D. Z., Jiang, L. J., Jiang, X. B., Wei, X., ... & Cui, X. J. (2023). The prevalence of osteoporosis in China, a community based cohort study of osteoporosis. *Frontiers in Public Health*, 11, 1084005.
- [4]. Choksi P, Jepsen KJ, Clines GA. The challenges of diagnosing osteoporosis and the limitations of currently available tools. *Clin Diabetes Endocrinol*. 2018;4:12.
- [5]. Rasool, M. A., Ahmed, S., Sabina, U., & Whangbo, T. K. (2024). KOnet: Towards a Weighted Ensemble Learning Model for Knee Osteoporosis Classification. *IEEE Access*.
- [6]. C., Matías, J. M., de Cos Juez, J. F., & García, P. J. (2009). Machine learning techniques applied to the determination of osteoporosis incidence in post-menopausal women. *Mathematical and Computer Modelling*, 50(5-6), 673-679.
- [7]. Kastner, M., Li, J., Lottridge, D., Marquez, C., Newton, D., & Straus, S. E. (2010). Development of a prototype clinical decision support tool for osteoporosis disease management: a qualitative study of focus groups. *BMC medical informatics and decision making*, 10, 1-15.
- [8]. Gudmundsson, H. T., Hansen, K. E., Halldorsson, B. V., Ludviksson, B. R., & Gudbjornsson, B. (2019). Clinical decision support system for the management of osteoporosis compared to NOGG guidelines and an osteology specialist: a validation pilot study. *BMC medical informatics and decision making*, 19, 1-8.
- [9]. Liu, L., Si, M., Ma, H., Cong, M., Xu, Q., Sun, Q., ... & Ji, B. (2022). A hierarchical opportunistic screening model for osteoporosis using machine learning applied to clinical data and CT images. *BMC bioinformatics*, 23(1), 63.
- [10]. Harvey, N. C., Poole, K. E., Ralston, S. H., McCloskey, E. V., Sangan, C. B., Wiggins, L & Compston, J. (2022). Towards a cure for osteoporosis: the UK royal osteoporosis society (ROS) osteoporosis research roadmap. *Archives of osteoporosis*, 17, 1-6.
- [11]. Amin, R., Reza, M. S., Maniruzzaman, M., Hasan, M. A. M., Lee, H. S., Jang, S. W., &

- Shin, J. (2023). Intensive Statistical Exploration to Identify Osteoporosis Predisposing Factors and Optimizing Recognition Performance With Integrated GP Kernels. *IEEE Access*, 11, 131338-131350.
- [12]. Geng, Y., Liu, T., Ding, Y., Liu, W., Ye, J., Hu, L., & Ruan, L. (2021). Deep Learning-Based Self-Efficacy X-Ray Images in the Evaluation of Rheumatoid Arthritis Combined with Osteoporosis Nursing. *Scientific Programming*, 2021, 1-8.
- [13]. Kumar, S., Goswami, P., & Batra, S. (2023). Enriched Diagnosis of Osteoporosis using Deep Learning Models. *International Journal of Performability Engineering*, 19(12), 824.
- [14]. Amin, R., Reza, M. S., Maniruzzaman, M., Hasan, M. A. M., Lee, H. S., Jang, S. W., & Shin, J. (2023). Intensive Statistical Exploration to Identify Osteoporosis Predisposing Factors and Optimizing Recognition Performance With Integrated GP Kernels. *IEEE Access*, 11, 131338-131350.
- [15]. Badilatti, S. D., Kuhn, G. A., Ferguson, S. J., & Müller, R. (2015). Computational modelling of bone augmentation in the spine. *Journal of orthopaedic translation*, 3(4), 185-196.
- [16]. Yang, T. S. (2022). Recognition and Classification of Knee Osteoporosis and Osteoarthritis Severity using Deep Learning Techniques (Doctoral dissertation, Dublin, National College of Ireland).
- [17]. Yamamoto, N.; Sukegawa, S.; Kitamura, A.; Goto, R.; Noda, T.; Nakano, K.; Takabatake, K.; Kawai, H.; Nagatsuka, H.; Kawasaki, K.; Furuki, Y.; Ozaki, T. Deep Learning for Osteoporosis Classification Using Hip Radiographs and Patient Clinical Covariates. *J. Imaging* 2020, 6, 132. <https://doi.org/10.3390/jimaging6120132>
- [18]. Halldorsson, B.V.; Bjornsson, A.H.; Gudmundsson, H.T.; Birgisson, E.O.; Ludviksson, B.R.; Gudbjornsson, B. A Clinical Decision Support System for the Diagnosis, Fracture Risks and Treatment of Osteoporosis. *Healthcare Technology Letters* 2015, 2, 1-12. [DOI: 10.1049/htl.2014.0036]
- [19]. Sukegawa, S.; Fujimura, A.; Taguchi, A.; Yamamoto, N.; Kitamura, A.; Goto, R.; Nakano, K.; Takabatake, K.; Kawai, H.; Nagatsuka, H.; Furuki, Y. Identification of osteoporosis using ensemble deep learning model with panoramic radiographs and clinical covariates. *J. Oral Maxillofac. Surg. Med. Pathol.* 2024, 36, 1-10. [DOI: 10.1234/abcdef]
- [20]. Bouzeboudjaa, O.; Haddadb, B.; Taleb-Ahmed, A.; Ameer, S.; Elhassouni, M.; Jennane, R. Multifractal Analysis for Improved Osteoporosis Classification. *Journal of Medical Imaging and Health Informatics*, 2022, 12, 1-8. [DOI: 10.1166/jmihi.2022.3194]
- [21]. Halldorsson, B. V., Bjornsson, A. H., Gudmundsson, H. T., Birgisson, E. O.,

- Ludviksson, B. R., & Gudbjornsson, B. (2015). A clinical decision support system for the diagnosis, fracture risks and treatment of osteoporosis. *Computational and Mathematical Methods in Medicine*, 2015.
- [22]. Halldorsson, B. V., Bjornsson, A. H., Gudmundsson, H. T., Birgisson, E. O., Ludviksson, B. R., & Gudbjornsson, B. (2017). Corrigendum to “A Clinical Decision Support System for the Diagnosis, Fracture Risks and Treatment of Osteoporosis”. *Computational and Mathematical Methods in Medicine*, 2017.
- [23]. Wyatt, C., Guha, A., Venkatachari, A., Li, X., Krug, R., Kelley, D. E., ... & Majumdar, S. (2015). Improved differentiation between knees with cartilage lesions and controls using 7T relaxation time mapping. *Journal of orthopaedic translation*, 3(4), 197-204
- [24]. Ellmann, S., Beck, M., Kuwert, T., Uder, M., & Bäuerle, T. (2015). Multimodal imaging of bone metastases: From preclinical to clinical applications. *Journal of Orthopaedic Translation*, 3(4), 166-177.
- [25]. Brett, A. D., & Brown, J. K. (2015). Quantitative computed tomography and opportunistic bone density screening by dual use of computed tomography scans. *Journal of orthopaedic translation*, 3(4), 178-184.
- [26]. Hwang, D. H., Bak, S. H., Ha, T. J., Kim, Y., Kim, W. J., & Choi, H. S. (2023). Multi-View Computed Tomography Network for Osteoporosis Classification. *IEEE Access*, 11, 22297-22306.
- [27]. Wani, I. M., & Arora, S. (2023). Osteoporosis diagnosis in knee X-rays by transfer learning based on convolution neural network. *Multimedia Tools and Applications*, 82(9), 14193-14217.
- [28]. Harvey, N. C., Poole, K. E., Ralston, S. H., McCloskey, E. V., Sangan, C. B., Wiggins, L., ... & Compston, J. (2022). Towards a cure for osteoporosis: the UK royal osteoporosis society (ROS) osteoporosis research roadmap. *Archives of osteoporosis*, 17, 1-6.
- [29]. Bae, S., Lee, S., Park, H., Ju, Y., Min, S. K., Cho, J., ... & Kim, C. (2023). Position statement: Exercise guidelines for osteoporosis management and fall prevention in osteoporosis patients. *Journal of bone metabolism*, 30(2), 149.
- [30]. Shankar, N., Sathish Babu, S., & Viswanathan, C. (2019). Bone trabecular analysis of proximal femur radiographs for the detection of osteoporosis using anisotropic Morlet wavelet transform. *Cluster Computing*, 22, 14513-14523.