Health Monitoring and Disease Prediction using Deep learning Model

Sahithi Thulluru
700752037
Dept. Computer Science
University of Central Missouri
sxt20370@ucmo.edu

Sahil Naidu Pagadala
700748694
Dept. Computer Science
University of Central Missouri
sxp86940@ucmo.edu

Pavan Uppala
700754009
Dept. Computer Science
University of Central Missouri
pxu40090@ucmo.edu

Priyanka Nayudu
700756568
Dept. Computer Science
University of Central Missouri
pxn65680@ucmo.edu

***Abstract* -- Machine learning is the process of teaching computers to function as efficiently as possible by utilizing examples or historical data. It is the study of machine learning—the learning of systems from data and experience. Training and testing are the two main stages of a machine learning system. Predicting illnesses based on patient history and symptoms is one use of machine learning. With its many applications in the medical field, machine learning technology has been developing for decades and is now able to address many issues related to healthcare. Healthcare systems may more effectively analyse data and provide findings by utilizing machine learning. This helps physicians make better judgments about patient diagnoses and treatment options, which ultimately improves patient outcomes.**

Our goal in this study is to integrate deep learning neural networks—a cutting-edge machine learning capability—within a single, integrated healthcare system. Predictive machine learning algorithms for illness prediction can greatly improve healthcare decision-making in situations when a diagnosis is not immediately obvious. Deep learning neural networks are especially well-suited for jobs involving illness prediction because they are highly adept at deriving complicated patterns from unprocessed data. Healthcare systems can become more intelligent and capable of making accurate illness predictions even in cases when diseases are not detectable in their early stages by utilizing deep learning technology. This is consistent with the idea that "prevention is better than cure," as early illness preventive efforts can be facilitated by early disease prediction and outbreak prediction.

This document's main goal is to create a system for immediate medical treatment, which includes gathering symptoms from multisensorial devices, combining them with other medical data, and putting them in patient records. Subsequently, sophisticated deep learning neural network techniques would be employed to examine this dataset. Deep learning neural networks are better at deciphering intricate correlations in data, which enables more precise predictions and insights to be made in healthcare applications.

***Keywords* -- Simple Feedforward Neural Network, Symptoms, Health Care, Machine learning, Deep learning neural networks, Cross-Entropy Loss, Adam Optimizer, Heatmaps, ReLU, Softmax Activation, K-fold cross validation, Decision tree.**

I. INTRODUCTION

For decades, the healthcare industry has faced the challenge of accurately predicting diseases using patient treatment histories and health data. Traditional methods, such as data mining techniques, have been employed to forecast specific diseases by analysing pathology data or medical profiles. However, recent advancements in deep learning have brought about a paradigm shift in predictive analytics within healthcare. Deep learning neural networks, renowned for their ability to learn intricate patterns directly from raw data with minimal preprocessing, have emerged as powerful tools in processing vast and complex medical datasets. Unlike traditional approaches, deep learning models can derive rich, hierarchical representations of data, leading to more precise predictions.

The primary aim of integrating deep learning neural networks into healthcare is to enhance patient care and improve outcomes. By harnessing the capabilities of deep learning, healthcare professionals can streamline diagnosis processes and enhance treatment efficacy. Deep learning

https://github.com/Sahilnaidupagadala03/NN_Final_Project algorithms excel at extracting valuable insights from medical data, empowering clinicians to make informed decisions about patient diagnoses and treatment plans. This advanced technology has the potential to significantly impact healthcare delivery by enabling earlier disease detection, facilitating personalized treatment approaches, and ultimately, improving patient outcomes.

In the realm of healthcare, deep learning neural networks find application across a wide spectrum of tasks. These include disease prediction based on symptoms and medical histories, medical image analysis for diagnostic and therapeutic purposes, and the analysis of large-scale healthcare datasets for clinical insights. Through the utilization of deep learning, healthcare organizations can unlock the full potential of their data, leading to more accurate predictions and enhanced decision-making capabilities.

The integration of deep learning neural networks into healthcare signifies a transformative shift in how diseases are predicted and managed. The ability to predict diseases accurately and intervene early holds immense potential in improving treatment outcomes and saving lives. Despite the promising outlook, challenges persist, such as concerns regarding data privacy, interpretability of deep learning models, and the need for robust validation and regulatory approval. Addressing these challenges will be crucial for the widespread adoption of deep learning in healthcare. Moreover, future research endeavours should focus on developing more interpretable deep learning models and ensuring equitable access to advanced healthcare technologies.

In conclusion, deep learning neural networks have the capacity to revolutionize disease prediction and healthcare delivery. By leveraging advanced machine learning techniques, healthcare professionals can elevate patient care, treatment outcomes, and ultimately, contribute to saving lives. The integration of deep learning into healthcare represents a significant leap forward, holding great promise for the future of medicine.

II. MOTIVATION

Medical facilities must be improved in order to make better decisions about patient diagnosis and treatment alternatives. Deep learning in healthcare assists people in processing large and complicated medical information and analyzing them to derive therapeutic insights. This can then be used by doctors to provide medical care. As a result, when deep learning is used in healthcare, it can lead to higher patient satisfaction. In this study, we attempt to integrate deep learning functionality in healthcare into a single system. Instead of diagnosis, healthcare may be made smart by implementing disease prediction utilizing Deep learning predictive algorithms. Some circumstances may arise when an early diagnosis of an illness is not possible. As a result, disease prediction can be implemented efficiently. As widely said “Prevention is better than cure”, prediction of diseases and epidemic outbreak would lead to an early prevention of an occurrence of a disease.

III. OBJECTIVES

In this research endeavor, we harness the power of deep learning to analyze healthcare data, both structured and unstructured, aiming to assess the risk of illness with unprecedented accuracy and depth. Departing from traditional statistical methods, we delve into the realm of neural networks, leveraging their ability to learn intricate patterns and representations from complex datasets.

Structured data, such as medical records collected from hospitals, pose unique challenges, including missing data. To address this, we adopt a latent factor model approach within our neural network architecture. This enables us to reconstruct missing data, thereby enhancing the completeness and reliability of our analyses.

In parallel, we tackle unstructured text data, which often contains invaluable insights buried within its textual fabric. Here, we employ deep learning techniques to automatically identify features using approaches such as the c-means algorithm. By extracting meaningful features from unstructured text data, we enrich our analyses and gain a more comprehensive understanding of the underlying health dynamics.

Throughout our research journey, collaboration with domain experts, including hospital specialists, is paramount. Their insights guide us in navigating the intricacies of structured data, ensuring that our analyses capture the nuances inherent in healthcare datasets.

With our deep learning approach, we aim to achieve several objectives:

Accurately determine disease risk, leveraging both structured and unstructured data.

Empower medical students working in pathology labs with advanced predictive analytics, aiding in diagnostic processes and treatment planning.

Support nurse students by providing insights gleaned from deep learning analyses, enhancing their understanding of disease dynamics and patient care strategies.

Equip doctors with sophisticated tools for addressing disease-related health issues, enabling more informed decision-making and personalized treatment approaches.

IV. MAIN CONTRIBUTIONS

Data Preprocessing:

Data preprocessing is a crucial step in machine learning pipelines aimed at ensuring data quality and compatibility with the chosen model. In this context, the dataset is initially separated into features (x) and the target variable (y). This step helps to clearly delineate the independent variables from the dependent variable, facilitating subsequent analysis.

Normalization of features is then applied to standardize the scale across different features. Normalization helps in preventing features with larger scales from dominating the training process, ensuring fair treatment for all features. It involves scaling each feature to a range between 0 and 1, or to a mean of 0 and a standard deviation of 1, making the data more amenable to the learning process.

Model Selection (simple feedforward neural network):

The neural network architecture is defined within a class called NeuralNetwork, which inherits from nn.Module, a PyTorch module providing neural network functionality. The architecture typically consists of two fully connected layers (fc1 and fc2) employing rectified linear unit (ReLU) activation functions to introduce non-linearity into the model.

Additionally, dropout regularization is implemented to mitigate overfitting, a common issue in neural networks. Dropout randomly deactivates a fraction of neurons during training, forcing the network to learn more robust and generalized features.

Hyper parameter Tuning:

Hyperparameter tuning is a crucial aspect of optimizing the performance of a neural network model. Parameters such as learning rates and hidden layer sizes are defined upfront, and cross-validation using K-Fold is employed to evaluate the model's performance across different parameter combinations.

During cross-validation, the dataset is divided into k subsets, or folds, with each fold used as a validation set while the rest are used for training. This process is repeated k times, ensuring that each fold serves as both training and validation data, providing a robust estimate of model performance.

Training:

Within each fold of cross-validation, the neural network model is trained using the defined architecture and hyperparameters. Training is conducted using the Adam optimizer, a popular choice due to its adaptive learning rate properties, and the cross-entropy loss function, suitable for classification tasks.

Model Evaluation:

Following training, the best-performing model based on cross-validation accuracy is selected and evaluated on the full training set. Model accuracy is computed by comparing predicted labels with ground truth labels, providing insights into the model's performance on the training data.

Testing:

Finally, a test function is defined to make predictions on new data, represented by user-provided symptoms. These symptoms are converted into a binary input vector and passed through the trained neural network to predict the associated disease, enabling the model to generalize to unseen instances.

V. RELATED WORK

1. Paper Name: Disease phenol type similarity improves the prediction of novel disease associated micro RNAs Author: Duc-Hau Le[1]

Several studies have demonstrated the importance of miRNAs (microRNAs) in human disease, and a variety of computational approaches have been developed to anticipate such connections by ranking candidate microRNAs according to their relevance to a disease. Network-based techniques are gaining traction since they make efficient use of the "disease module" idea in functionally similar miRNA networks. RWRMDA is a state-of-the-art algorithm-based Random Walk with Restart (RWR) technique on a functional similarity network miRNA. Because the concept of "disease module" also exists in protein interaction networks, the application of this method was motivated by its performance in predicting disease genes. Several more algorithms have also been designed for the prediction of disease genes. Regardless, they have not been used yet for disease microRNA prediction

2. Paper Name: Defining Disease Phenotypes in Primary Care Electronic Health Records by a Machine Learning Approach: A Case Study in Identifying Rheumatoid Arthritis Author: Shang-Ming Zhou[2]

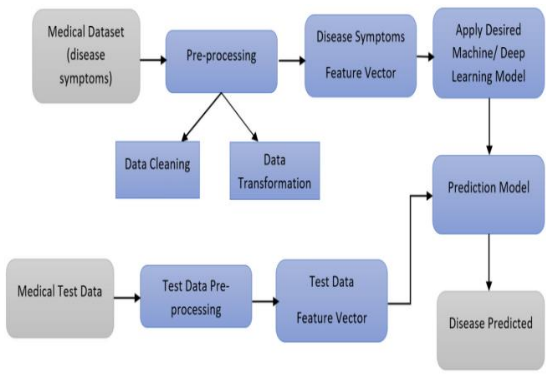
Variables were chosen by comparing the relative frequencies of Read codes in the primary care dataset associated with disease cases versus non-disease controls (disease/non-disease based on secondary care diagnosis); ii) predictors/associated variables were reduced using a Random Forest method; and iii) decision rules were inferred from decision tree models. The suggested method was then thoroughly verified on a separate dataset and its performance was compared to two existing deterministic algorithms for RA created with expert clinical knowledge.

3. Paper Name: Design and Implementing Heart Disease Prediction Using Navies Bayesian Author: Anjan Nikhil Repaka, Sai Deepak Ravikanti

The suggested method includes the following phases: dataset selection, application-based user registration and login, Bayesian Navy classification, prediction, and stable data transfer utilizing AES (Advanced Encryption Standard). The results are then generated. The work elaborates and offers many information abstraction strategies by utilizing data mining tools used for heart disease prediction. The findings suggest that the developed diagnostic framework accurately predicts risk factors for heart disease.[3]

VI. PROPOSED FRAMEWORK

I. Detail design



Data Preprocessing:

The dataset is split into features (x) and the target variable (y).

The features are normalized to ensure consistent scaling across different features.

Neural Network Definition:

A neural network class NeuralNetwork is defined, inheriting from nn.Module.

The architecture consists of two fully connected layers (fc1 and fc2) with ReLU activation and softmax output.

Dropout regularization is applied to prevent overfitting.

Hyperparameter Tuning and Cross-Validation:

Hyperparameters such as learning rates and hidden layer sizes are defined.

Cross-validation using K-Fold is performed to assess model performance.

The training loop iterates over different combinations of hyperparameters and evaluates model accuracy on validation data.

Training:

Within each fold of cross-validation, a neural network model is trained using the defined architecture and hyperparameters.

Training is performed using the Adam optimizer with cross-entropy loss.

Evaluation:

The best model, based on cross-validation accuracy, is selected and evaluated on the full training set.

Model accuracy is computed by comparing predicted labels with ground truth labels.

Testing:

A test function is defined to make predictions on new data (symptoms).

User-provided symptoms are converted into a binary input vector and passed through the trained neural network to predict the associated disease.

System configuration

This project may be run on standard hardware. We ran the entire project on an Intel I5 processor with 8 GB RAM and a 2 GB Nvidia Graphic Processor. It also has two cores that run at 1.7 GHz and 2.1 GHz. The first half of the process is the training phase, which takes about 10-15 minutes, and the second part is the testing phase, which just takes a few seconds to generate predictions and calculate accuracy.

Hardware Requirements:

• RAM: 4 GB

• Storage: 500 GB

• CPU: 2 GHz or faster

• Architecture: 32-bit or 64-bit

Software requirements

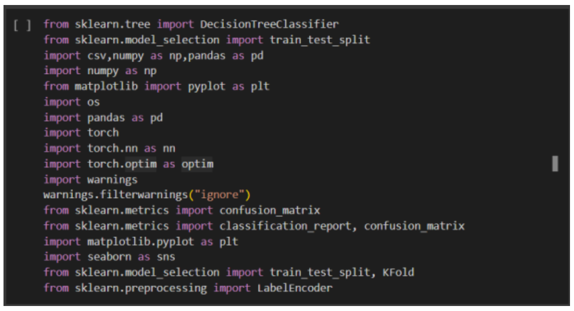
• Python 3.5 in Google Colab is used for data pre-processing, model training and prediction.

• Operating System: windows 7 and above or Linux based OS or MAC OS.

II. Implementation

Importing Necessary Libraries:

The sample below includes all of the necessary features for machine learning, data analysis, and neural network training. The first step is loading the necessary libraries, which include PyTorch for neural network training, NumPy and Pandas for data processing, Matplotlib and Seaborn for visualization, and scikit-learn for machine learning tasks. The code also contains instructions on how to silence warnings so that the runtime output is clearer. The next step involves importing evaluation metrics and visualization tools, which make it easier to examine the success of the model. Examples of these functions are the confusion matrix and classification report from scikit-learn, along with visualization libraries for making visually appealing plots. In order to prepare data for model training, the snippet imports modules for data preparation activities, which include dividing the dataset into training and testing subsets and encoding categorical variables. Together, these components form a comprehensive set of functionalities for data analysis, machine learning, and neural network training, laying the groundwork for building and evaluating predictive models effectively.



```
[ ] from sklearn.tree import DecisionTreeClassifier
    from sklearn.model_selection import train_test_split
    import csv,numpy as np,pandas as pd
    import numpy as np
    from matplotlib import pyplot as plt
    import os
    import pandas as pd
    import torch
    import torch.nn as nn
    import torch.optim as optim
    import warnings
    warnings.filterwarnings("ignore")
    from sklearn.metrics import confusion_matrix
    from sklearn.metrics import classification_report, confusion_matrix
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split, KFold
    from sklearn.preprocessing import LabelEncoder
```

Data Loading and Preprocessing

Loading a CSV file named 'Training.csv' using Pandas' **read_csv** function and encodes it using UTF-8. The data is then converted into a DataFrame named 'df'. The features and target variable are extracted from the DataFrame: 'x' contains all columns except the last one (features), and 'y' contains the last column ('prognosis') which represents the target variable.

Data Splitting

Here the dataset is split into training and testing sets using the train_test_split function from scikit-learn. It assigns 67% of the data to the training set ('X_train' and 'y_train') and 33% to the testing set ('X_test' and 'y_test'). The 'random_state' parameter is set to 42 for reproducibility. Subsequently, it prints the counts of samples and labels in both the training and testing sets for verification purposes.

Correlation Analysis and Visualization

Heatmaps:

Heatmaps are graphical data visualizations that show the magnitude of each value in a matrix by using colours to symbolize the values. In order to identify patterns and correlations within datasets, they are frequently employed in data analysis and visualization, especially in correlation analysis.

Seaborn's Heatmap Function:

Developed on top of Matplotlib, Seaborn is a Python data visualization library. It offers a sophisticated interface for making eye-catching and educational statistical visuals. Heatmap visualizations are specifically intended to be produced by Seaborn's heatmap function. It draws a grid of coloured squares, each of which represents a value in the dataset, using an input that resembles a matrix. Each square's colour intensity represents the magnitude of the value, making it simple to see patterns and trends in the data.

Correlation Analysis:

Correlation analysis is a statistical technique used to measure the strength and direction of the linear relationship between two variables. In the context of heatmap visualization, correlation analysis is typically applied to numerical datasets to explore the relationships between different variables. The correlation coefficient, ranging from -1 to 1, quantifies the strength and direction of the relationship:

- A value close to 1 indicates a strong positive correlation (as one variable increases, the other also tends to increase).
- A value close to -1 indicates a strong negative correlation (as one variable increases, the other tends to decrease).
- A value close to 0 indicates little to no linear correlation between the variables.

Benefits of Heatmap Visualization:

- **Visual Clarity**: Heatmaps provide a clear and intuitive visualization of complex datasets, making it easy to identify relationships and patterns.
- **Efficient Communication**: Heatmaps convey information effectively, making them valuable tools for communicating findings to stakeholders and decision-makers.
- **Insight Generation**: By visualizing correlations between variables, heatmaps facilitate the discovery of insights and actionable recommendations for further analysis or decision-making.

Overall, correlation analysis and visualizes the correlations between numerical features in the dataset. It first selects only the numerical columns from the DataFrame using the **select_dtypes** method with **include=[np.number]**. Then, a heatmap is generated using Seaborn's **heatmap** function, displaying the correlation matrix of the numerical features. Each cell in the heatmap represents the correlation coefficient between two features, and the color intensity indicates the strength of the correlation. Additionally, annotations are included within the heatmap to display the correlation values. Finally, the plot is displayed using Matplotlib. Adjustments to the font size of the annotations are made for better readability. This visualization helps in understanding the relationships between different numerical features in the dataset, which can be useful for feature selection or identifying multicollinearity issues in predictive modeling tasks.

Scatter Matrix Plot and Histogram Visualization

Python function **plotScatterMatrix** designed to generate a scatter matrix plot along with kernel density estimations (KDE) and histograms for numerical features in a DataFrame. The function takes the following parameters:

- **df**: The DataFrame containing numerical features to be visualized.
- **plotSize** (optional): The size of the scatter matrix plot (default: 20).
- **textSize** (optional): The font size of annotations (default: 10).

plotScatterMatrix function provides a powerful tool for visualizing and understanding the relationships between numerical features in a DataFrame, enabling data scientists and analysts to gain insights and make informed decisions based on the data's characteristics.

Neural Network Training and Evaluation

The neural network being implemented here is a simple feedforward neural network with one hidden layer. Specifically, it consists of an input layer, one hidden layer with ReLU activation function, and an output layer with softmax activation function. This type of architecture is commonly used for multi-class classification tasks. The input features are fed into the neural network, processed through the hidden layer to extract features, and then transformed into class probabilities at the output layer using softmax activation. The model is trained using the CrossEntropyLoss criterion and optimized using the Adam optimizer. Overall, it's a basic implementation of a feedforward neural network for classification purposes.

Training a neural network for multi-class classification involves several steps, including data preparation, model architecture definition, hyperparameter tuning, model training, and evaluation. By following these steps and optimizing the model's parameters, one can develop an effective classifier capable of accurately predicting class labels for input data.

Simple Feedforward Neural Network:

A feedforward neural network, also known as a multilayer perceptron (MLP), is a fundamental type of artificial neural network where information flows in one direction, from input to output. It consists of multiple layers of interconnected neurons (nodes), organized into an input layer, one or more hidden layers, and an output layer. Each neuron in the network receives input signals, performs a weighted sum of these inputs, applies an activation function to produce an output, and passes it to the neurons in the next layer. Simple feedforward neural networks are widely used for tasks such as classification, regression, and function approximation.

Cross-Entropy Loss Criterion:

Cross-entropy loss, also known as log loss, is a commonly used loss function in classification tasks, particularly in the context of multi-class classification. It measures the difference between the predicted probability distribution and the actual distribution of class labels in the training data. Cross-entropy loss penalizes incorrect predictions more severely when the predicted probability diverges from the true distribution. Mathematically, it is defined as the negative log likelihood of the observed data given the predicted probabilities. Minimizing cross-entropy loss encourages the model to produce probability distributions that are closer to the ground truth labels.

Adam Optimizer:

Adam (Adaptive Moment Estimation) is an adaptive learning rate optimization algorithm commonly used to train neural networks. It combines the advantages of two other popular optimization techniques: AdaGrad, which adapts the learning rates of model parameters based on their past gradients, and RMSprop, which uses exponentially weighted moving averages of squared gradients to adjust the learning rates. Adam maintains separate adaptive learning rates for each parameter, allowing it to converge quickly and efficiently on a wide range of optimization problems. It is particularly effective for training deep neural networks and is widely used in practice due to its robust performance and ease of use.

Data Preparation:

The dataset is split into features (input data) and target variable (labels). It's essential to preprocess the data, including handling missing values, encoding categorical variables, and normalizing numerical features to ensure effective training.

Encoding Target Labels:

If the target variable contains categorical labels, they need to be encoded into numerical format suitable for training a neural network. This typically involves techniques like one-hot encoding or label encoding.

Model Architecture Definition:

Define the architecture of the neural network model using frameworks like PyTorch or TensorFlow. Specify the number of input features, hidden layers, activation functions, and output layer structure suitable for the classification task.

Hyperparameter Tuning:

Choose appropriate hyperparameters for the neural network model, including learning rate, batch size, number of epochs, and optimizer (e.g., Adam, SGD). Hyperparameter tuning is crucial for optimizing model performance and preventing overfitting.

Model Initialization and Optimization:

Initialize the neural network model with the defined architecture. Specify the loss function suitable for multi-class classification tasks, such as cross-entropy loss. Select an optimizer algorithm to update the model parameters based on the gradients computed during training.

Training Loop:

Train the neural network model on the training data using a training loop. In each epoch, perform a forward pass to compute the predicted outputs, calculate the loss using the specified loss function, and perform a backward pass to compute gradients and update the model parameters using the optimizer.

Evaluation:

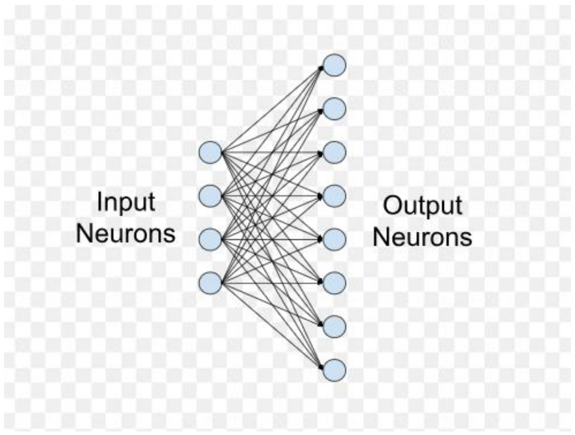
After training, evaluate the trained model's performance on a separate validation or test dataset. Compute evaluation metrics such as accuracy, precision, recall, and F1-score to assess the model's effectiveness in classifying unseen data.

Neural Network Training for Multi-Class Classification

Training a neural network for multi-class classification involves several steps, including data preparation, model architecture definition, hyperparameter tuning, model training, and evaluation. By following these steps and optimizing the model's parameters, one can develop an effective classifier capable of accurately predicting class labels for input data.

Fully Connected (Linear) Layers:

Fully connected layers, also known as dense layers, are fundamental building blocks in neural networks. Each neuron in a fully connected layer is connected to every neuron in the preceding layer, forming a dense network of connections. These connections are represented by weights, which are learnable parameters adjusted during the training process to minimize the loss function.



In the context of a neural network architecture, fully connected layers perform linear transformations of the input data. Each neuron in the layer computes a weighted sum of the inputs, including the bias term, and applies an activation function to the result. This process allows the network to learn complex nonlinear relationships between features in the data.

ReLU Activation Function:

ReLU, which stands for Rectified Linear Unit, is a popular activation function used in neural networks. It introduces nonlinearity into the model by outputting the input directly if it is positive, and zero otherwise. Mathematically, ReLU is defined as:

$$f(x) = \max(0, x)$$

ReLU activation function is preferred over traditional activation functions like sigmoid and tanh due to its simplicity and effectiveness in preventing the vanishing gradient problem during training. It helps the network learn faster and reduces the likelihood of saturation in the gradients, leading to more efficient training and better convergence.

Softmax Activation Function:

Softmax is a commonly used activation function in the output layer of neural networks for multi-class classification tasks. It transforms the raw output scores (logits) of the neural network into probabilities, representing the likelihood of each class.

Softmax function computes the probability distribution over multiple classes by exponentiating each output score and normalizing them by the sum of all exponentiated scores. Mathematically, softmax is defined as:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

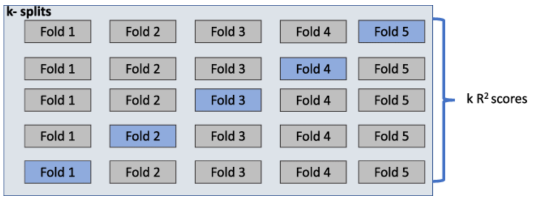
Softmax ensures that the output probabilities sum up to 1, making it suitable for classification problems where the model needs to predict the probability of each class. The class with the highest probability is chosen as the predicted class label.

Following it, a sample data point is prepared for inference by converting it into a numpy array and normalizing it based on statistics computed from the training data. The normalized data is then converted into a PyTorch tensor for compatibility with the model. Predictions are obtained by passing the tensor through the model, and the predicted class labels are extracted using the inverse transformation provided by the label encoder. Additionally, the code defines a function to predict symptoms based on user input. This function initializes a label representing symptoms, converts it into a tensor, makes predictions using the model, and returns the predicted symptom. Overall, the code illustrates a general workflow for making predictions with a trained neural network model. Later performing the Hyperparameter tuning over different learning rates and hidden layers using K-fold cross validation. Early stopping is implemented based on validation loss to prevent overfitting. L2 regularization is added to the optimizer for further regularization. The best model based on cross-validation accuracy is evaluated on the full training set.

Later we also check using Decision tree where we acquire 100% accuracy.

K-fold cross validation

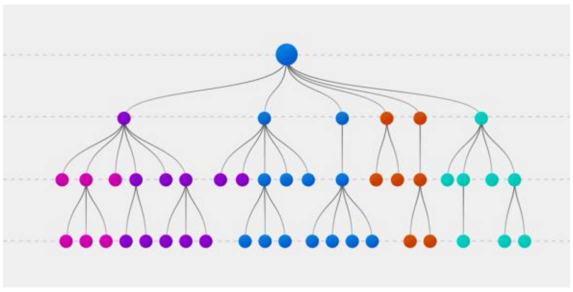
K-fold cross-validation is a resampling technique used to assess the performance of a machine learning model. It involves partitioning the dataset into K subsets, or "folds," of approximately equal size. The model is trained K times, each time using K-1 folds as training data and one-fold as validation data. The performance of the model is then evaluated by averaging the results from the K iterations. K-fold cross-validation helps to ensure that the model's performance estimates are less dependent on the particular data partitioning, providing a more reliable evaluation of its generalization ability. It is commonly used for hyperparameter tuning and model selection in machine learning tasks.



Usually used to detect overfitting, i.e., failing to generalize a pattern.

Decision tree

A decision tree is a supervised machine learning algorithm used for both classification and regression tasks. It operates by recursively partitioning the feature space into smaller regions based on the values of input features, creating a tree-like structure where each internal node represents a feature and each leaf node corresponds to a class label (for classification) or a predicted value (for regression). The algorithm selects the best split at each node based on criteria such as Gini impurity or information gain, aiming to maximize homogeneity within each resulting subset. Decision trees are intuitive, easy to interpret, and capable of handling both numerical and categorical data.



VII. DATA DESCRIPTION

A dataset to provide the students a source to create a healthcare related system. Get dummies processed file will be available at

https://www.kaggle.com/rabisingh/symptom-checker?select=Training.csv

This dataset comprises columns representing diseases, symptoms, precautions, and associated weights. Cleaning the dataset can be facilitated through file handling in any programming language. Understanding the arrangement of rows and columns is essential for effective data cleaning. By grasping the organization of data within the dataset, users can utilize file handling techniques to perform tasks such as removing duplicate entries, handling missing values, and ensuring uniform formatting. This process enhances data quality and reliability, enabling accurate analysis and insights extraction. Overall, comprehension of the dataset's structure empowers users to efficiently clean and prepare the data for further analysis or modelling purposes.

VIII. RESULT ANALYSIS

Data visualization plays a crucial role in interpreting and communicating findings effectively. Python offers a variety of powerful plotting libraries, each with distinct strengths tailored to different needs.

Matplotlib: Known for its flexibility, Matplotlib allows for fine-grained control over plot customization. It's ideal for creating a wide range of static plots, offering extensive functionality for customizing plot elements.

Pandas Visualization: Built on top of Matplotlib, Pandas Visualization provides a simplified interface for generating plots directly from Pandas DataFrame objects. It's particularly useful for quick exploratory data analysis and plotting simple visualizations with minimal code.

Seaborn: Designed for statistical data visualization, Seaborn offers a high-level interface with aesthetically pleasing default styles. It simplifies the creation of complex visualizations, including heatmaps, violin plots, and pair plots, while also providing support for statistical estimation.

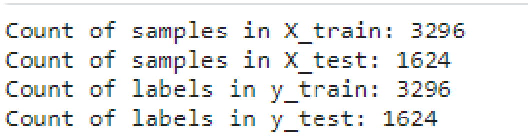
Plotnine: Inspired by R's ggplot2, Plotnine follows the Grammar of Graphics principles. It allows users to create declarative plots by mapping data variables to plot aesthetics, making it intuitive for users familiar with ggplot2 syntax.

Plotly: Notably, Plotly stands out for its ability to create interactive plots, which can be embedded into web applications or Jupyter Notebooks. It offers a wide range of chart types and customization options, enabling the creation of dynamic visualizations for data exploration and presentation.

<u>Preliminary Results</u>

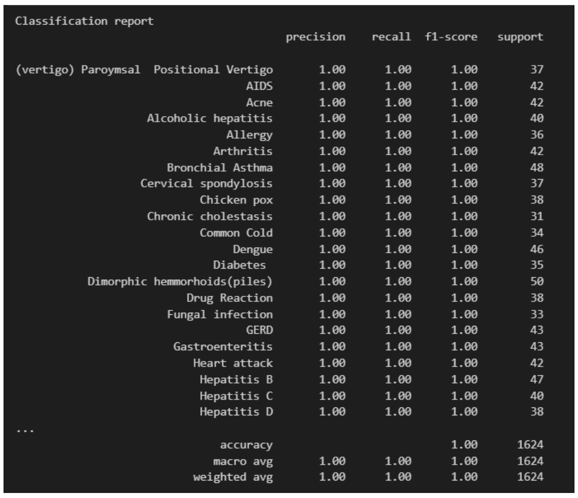
Data Stats

It is necessary to separate the available data into two independent sets, a training set and a testing set, in order to appropriately assess the performance of a machine learning model. This procedure guarantees that the model is assessed on an unknown set of data after being trained on one. The main objective is to replicate the model's performance on fresh, untested data. The testing set gets 33% of the data, while the remaining 67% will be used for training. The counts of samples and labels in each subgroup are indicated in the printed outputs, which offer insights into the distribution of samples between the training and testing sets. Understanding data partitioning and maintaining a suitable balance between training and testing data are critical for robust model assessment and generalization, and this phase is critical to achieving these goals.



```
Count of samples in X_train: 3296
Count of samples in X_test: 1624
Count of labels in y_train: 3296
Count of labels in y_test: 1624
```

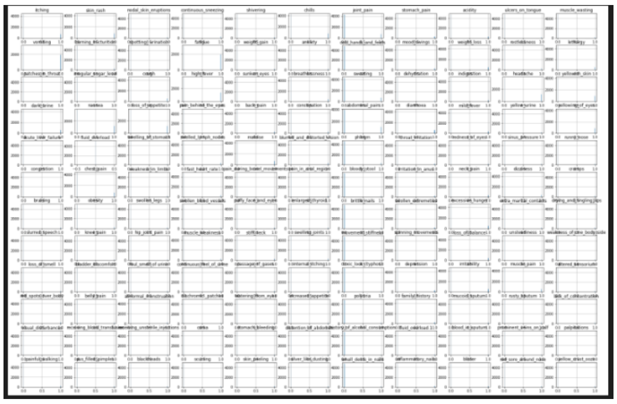
Classification Report (Hyper parameter Tuning)



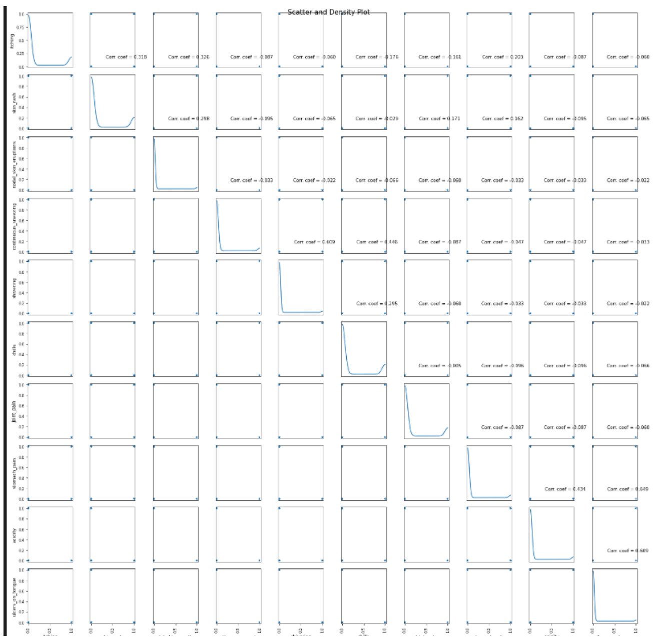
```
Classification report
                                         precision    recall  f1-score   support

(vertigo) Paroymsal  Positional Vertigo       1.00      1.00      1.00        37
                                   AIDS       1.00      1.00      1.00        42
                                   Acne       1.00      1.00      1.00        42
                    Alcoholic hepatitis       1.00      1.00      1.00        40
                                Allergy       1.00      1.00      1.00        36
                              Arthritis       1.00      1.00      1.00        42
                       Bronchial Asthma       1.00      1.00      1.00        48
                   Cervical spondylosis       1.00      1.00      1.00        37
                            Chicken pox       1.00      1.00      1.00        38
                    Chronic cholestasis       1.00      1.00      1.00        31
                            Common Cold       1.00      1.00      1.00        34
                                 Dengue       1.00      1.00      1.00        46
                              Diabetes        1.00      1.00      1.00        35
        Dimorphic hemmorhoids(piles)       1.00      1.00      1.00        50
                          Drug Reaction       1.00      1.00      1.00        38
                       Fungal infection       1.00      1.00      1.00        33
                                   GERD       1.00      1.00      1.00        43
                      Gastroenteritis       1.00      1.00      1.00        43
                           Heart attack       1.00      1.00      1.00        42
                            Hepatitis B       1.00      1.00      1.00        47
                            Hepatitis C       1.00      1.00      1.00        40
                            Hepatitis D       1.00      1.00      1.00        38
...
                               accuracy                           1.00      1624
                              macro avg       1.00      1.00      1.00      1624
                           weighted avg       1.00      1.00      1.00      1624
```

Histogram for the set:

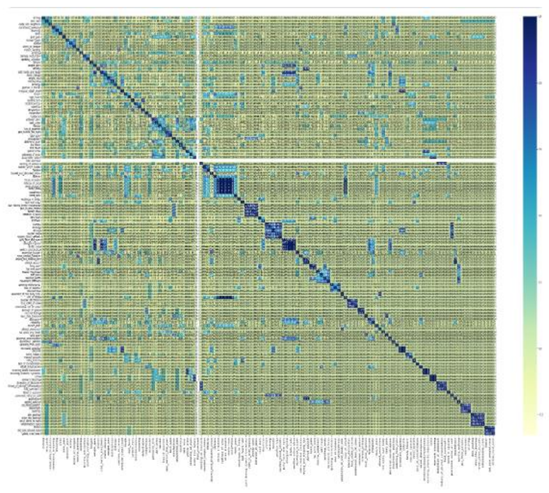


Scattered Plot



Correlation graph

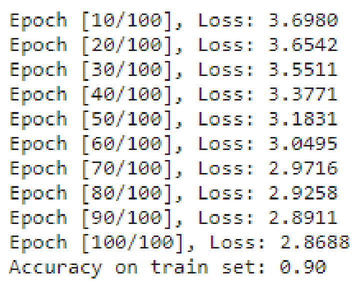
Visualizing the correlation matrix of the dataset provides valuable insights into the relationships between different variables. In the provided code snippet, a heatmap is generated using the sns.heatmap() function from the Seaborn library (sns), which displays the correlation coefficients between pairs of variables. Data analysts and machine learning professionals can find patterns and relationships in the information by viewing the correlation matrix, which helps with feature selection and model construction. In exploratory data analysis, this visual aid is essential for comprehending the underlying connections between variables.



Neural network

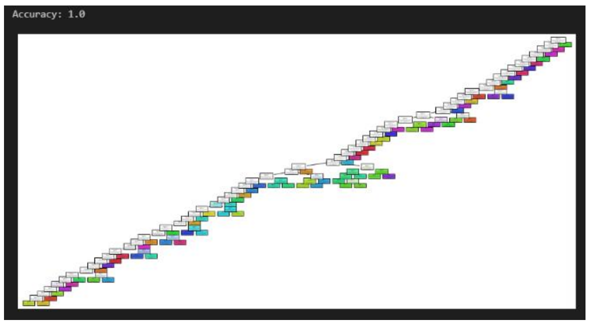
https://github.com/Sahilnaidupagadala03/NN_Final_Project

The use of PyTorch to create a neural network categorization model. Data preparation, which includes label encoding and feature normalization, comes first. The architecture of the neural network is established. It consists of two fully connected layers with ReLU activation and a softmax output layer in between. The Adam optimizer with cross-entropy loss is used for training, and it runs for 100 epochs. 90% accuracy is obtained while evaluating the model on the training set, demonstrating the model's ability to accurately predict the target variable using the given characteristics.



```
Epoch [10/100], Loss: 3.6980
Epoch [20/100], Loss: 3.6542
Epoch [30/100], Loss: 3.5511
Epoch [40/100], Loss: 3.3771
Epoch [50/100], Loss: 3.1831
Epoch [60/100], Loss: 3.0495
Epoch [70/100], Loss: 2.9716
Epoch [80/100], Loss: 2.9258
Epoch [90/100], Loss: 2.8911
Epoch [100/100], Loss: 2.8688
Accuracy on train set: 0.90
```

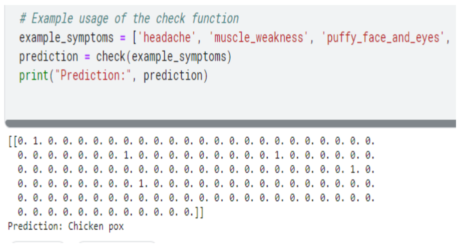
Decision Tree



```
Accuracy: 1.0
```

With Input

The check function plays a crucial role in the diagnosis of medical disorders. It is especially helpful in situations when patients report symptoms for first evaluation. The function first initializes a binary array to indicate whether each symptom is present or absent before methodically processing the input, which is a list of symptoms. After mapping symptoms to their associated indices using a specified vocabulary, the program uses a trained decision tree classifier to determine the most likely medical diagnosis. In one recent case, the diagnosis of chicken pox was made based on symptoms that included headache, muscular weakness, puffiness around the eyes and cheeks, slight temperature, and skin rash.



```
# Example usage of the check function
example_symptoms = ['headache', 'muscle_weakness', 'puffy_face_and_eyes',
prediction = check(example_symptoms)
print("Prediction:", prediction)
```

```
[[0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
Prediction: Chicken pox
```

This predictive capability enables quick identification and preliminary assessment of health concerns based on reported symptoms, facilitating timely intervention and appropriate medical guidance.

References/Bibliography

[1] "2015 2nd National Foundation for Science and Technology Development Conference on Information and Computer Science" Duc-Hau Le, "Disease phenotypic similarity improves the prediction of novel diseaseassociated microRNAs"

[2] Shang-Ming Zhou, "Defining Disease Phenotypes in Primary Care Electronic Health Records by a Machine Learning Approach: A Case Study in Identifying Rheumatoid Arthritis", doi:10.1371/journal.pone.0154515"

[3] Anjan Nikhil Repaka, "Design and Implementing Heart Disease Prediction Using Naives Bayesian," in Proceedings of the Third International Conference on Trends in Electronics and Informatics (ICOEI 2019); IEEE Xplore Part Number: CFP19J32-ART; ISBN: 978-1-5386-9439-8

[4] Min Chen, Yixue Hao, Kai Hwang, Fellow, IEEE, Lu Wang, and Lin Wang "Disease Prediction by Machine Learning from Healthcare Communities" (2017).

[5] Mr. Chala Beyene and Prof. Pooja Kamat, "Survey on Prediction and Analysis of Heart Disease Occurrence Using Data Mining Techniques," International Journal of Pure and Applied Mathematics, 2018.

[6] P. Groves, B. Kayyali, D. Knott, and S. V. Kuiken, 2016. "The 'big data' revolution in healthcare: Accelerating value and innovation."

[7] S. Patel and H. Patel, "Survey of data mining techniques utilised in the healthcare domain," International Journal of Informatics Science and Technology, Vol. 6, pp. 53-60, March 2016.

https://github.com/Sahilnaidupagadala03/NN_Final_Project

[8]. R. Banoth, A. K. Godishala, V. R and H. Yassin, "A Healthcare Monitoring System for Predicting Heart Disease through Recurrent Neural Network," 2022 IEEE 7th International conference for Convergence in Technology (I2CT), Mumbai, India, 2022, pp. 1-7, doi: 10.1109/I2CT54291.2022.9824888. keywords: {Heart; Recurrent neural networks; Databases; Soft sensors; Medical services; Machine learning; Predictive models; Recurrent Neural Network (RNN); Long Short-Term Memory (LSTM); Modified Artificial Flora Optimization (MAFO)},

[9]. I. Pavan Kumar, R. Mahaveer Kannan, K. Praveen Kumar, I. Basu, T. C. Anil Kumar and M. Choche, "A Design of Disease Diagnosis based Smart Healthcare Model using Deep Learning Technique," 2022 International Conference on Electronics and Renewable Systems (ICEARS), Tuticorin, India, 2022, pp. 1444-1449, doi: 10.1109/ICEARS53579.2022.9752063. keywords: {Simulation; Predictive models; Generative adversarial networks; Prediction algorithms; Real-time systems; Data models; Diabetes; Artificial Intelligence; Internet of Things; Deep Learning; Generative Adversarial Network; Smart Healthcare System; Disease Detection}

[10]. S. Hlawa and N. B. Romdhane, "Deep learning-based Alzheimer's disease prediction for smart health system," The 3rd International Conference on Distributed Sensing and Intelligent Systems (ICDSIS 2022), Hybrid Conference, Sharjah, United Arab Emirates, 2022, pp. 128-137, doi: 10.1049/icp.2022.2427.

[11]. Géron, Aurélien (2019). Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow. Sebastopol, CA: O'Reilly Media.

[12]. Zhang, Y. *et al.* Computational analysis and prediction of lysine malonylation sites by exploiting informative features in an integrative machine-learning framework. *Brief. Bioinform.* 20, 2185–2199 (2019).

[13]. Kwon, K., Kim, D. & Park, H. A parallel MR imaging method using multilayer perceptron. *Med. Phys.* 44, 6209–6224 (2017).

[14]. Tang, J., Deng, C. & Huang, G. B. Extreme learning machine for multilayer perceptron. *IEEE Trans. Neural Netw. Learn. Syst.* 27, 809–821 (2016).

[15]. Aashish Bansal, M. K. Nallakaruppan, Feslin Anish Mon, Veena Grover, Balamurugan Balusamy, "Heart Disease Prediction with Hyperparameter Analysis", *2023 International Conference on Computer Science and Emerging Technologies (CSET)*, pp.1-6, 2023.

[16]. Mehanas Shahul, Pushpalatha K. P, "Machine Learning Based Patient Classification In Emergency Department", *2023 International Conference on Advances in Intelligent Computing and Applications (AICAPS)*, pp.1-5, 2023.

[17]. Pabitha C, Kalpana V, Evangelin Sonia SV, Pushpalatha A, Mahendran G, Sivarajan S, "Development and Implementation of an Intelligent Health Monitoring System using IoT and Advanced Machine Learning Techniques", *Journal of Machine and Computing*, pp.456, 2023.

[18]. Karan Pal, Sarthak Panwar, Deepjyoti Choudhury, "A Pragmatic Approach of Heart and Liver Disease Prediction using Machine Learning Classifiers", *2024 International Conference on Emerging Systems and Intelligent Computing (ESIC)*, pp.728-734, 2024.

[19]. Captain Sukchayanan, Sujitra Arwatchananukul, Punnarumol Temdee, "Multi-Class Classification of Metabolic Syndrome Group Using Gradient Boosting", *2023 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON)*, pp.564-567, 2023.

[20]. Pierre Claver Bizimana, Zuping Zhang, Muhammad Asim, Ahmed A. Abd El-Latif, Mohamed Hammad, "Learning-based techniques for heart disease prediction: a survey of models and performance metrics", *Multimedia Tools and Applications*, 2023.