MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE GWALIOR
(A Govt. Aided UGC Autonomous Institute Affiliated to RGPV, Bhopal)
NAAC Accredited with A++ Grade

Project Report

on

Disease Prediction and Medical assistance using Machine Learning

Submitted By:

Umesh Patidar (0901AM211061)

Yashdeep Singh (0901AM211066)

Faculty Mentor:

Dr. Kritika Bansal

Assistant Professor

CENTRE FOR ARTIFICIAL INTELLIGENCE
MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE
GWALIOR - 474005 (MP) est. 1957

JULY-DEC. 2023

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE GWALIOR

(A Govt. Aided UGC Autonomous Institute Affiliated to RGPV, Bhopal)

NAAC Accredited with A++ Grade

CERTIFICATE

This is certified that **Umesh Patidar** (0901AM211061) and **Yashdeep Singh** (0901AM211066) has submitted the project report titled **Disease Prediction and Medical assistance using Machine Learning** under the mentorship of **Dr. Kritika Bansal**, in partial fulfilment of the requirement for the award of degree of Bachelor of Technology in **Artificial Intelligence and Machine Learning** from Madhav Institute of Technology and Science, Gwalior.

Kritika 22/11/2023

Dr. Kritika Bansal
Assistant Professor
Centre for Artificial Intelligence

23/11/22

Dr. R. R. Singh
Coordinator
Centre for Artificial Intelligence

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE GWALIOR

(A Govt. Aided UGC Autonomous Institute Affiliated to RGPV, Bhopal)

NAAC Accredited with A++ Grade

DECLARATION

I hereby declare that the work being presented in this project report, for the partial fulfilment of requirement for the award of the degree of Bachelor of Technology in **Artificial Intelligence and Machine Learning** at Madhav Institute of Technology & Science, Gwalior is an authenticated and original record of my work under the mentorship of **Dr. Kritika Bansal**, Assistance Professor, Centre for Artificial Intelligence

I declare that ! have not submitted the matter embodied in this report for the award of any degree or diploma anywhere else.

Umesh

Umesh Patidar
(0901AM211061)
Yashdeep Singh
(0901AM211066)
3rd Year,
Centre for Artificial Intelligence

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE GWALIOR

(A Govt. Aided UGC Autonomous Institute Affiliated to RGPV, Bhopal)

NAAC Accredited with A++ Grade

ACKNOWLEDGEMENT

The full semester project has proved to be pivotal to my career. I am thankful to my institute, **Madhav Institute of Technology and Science** to allow me to continue my disciplinary/interdisciplinary project as a curriculum requirement, under the provisions of the Flexible Curriculum Scheme (based on the AICTE Model Curriculum 2018), approved by the Academic Council of the institute. I extend my gratitude to the Director of the institute, **Dr. R. K. Pandit** and Dean Academics, **Dr. Manjaree Pandit** for this.

I would sincerely like to thank my department, **Centre for Artificial Intelligence,** for allowing me to explore this project. I humbly thank **Dr. R. R. Singh**, Coordinator, Centre for Artificial Intelligence, for his continued support during the course of this engagement, which eased the process and formalities involved.

I am sincerely thankful to my faculty mentors. I am grateful to the guidance of **Dr. Kritika Bansal**, Assistance Professor, Centre for Artificial Intelligence, for his continued support and guidance throughout the project. I am also very thankful to the faculty and staff of the department.

Umesh Patidar
(0901AM211061)
Yashdeep Singh
(0901AM211066)
3rd Year,
Centre for Artificial Intelligence

ABSTRACT

Disease Prediction using Machine Learning is a predictive modelling system designed to anticipate the occurrence of diseases based on symptoms provided by patients or users. The system employs a KNN classifier, Decision tree and Random Forest, to calculate the probability of a specific disease. This approach involves processing user-input symptoms to generate a probability output associated with potential diseases. Moreover, the model provides medical assistance tailored to the predicted disease.

As the volume of biomedical and healthcare data continues to increase, the accurate analysis of medical information becomes imperative for early disease detection and optimal patient care. In this context, the integration of linear regression and decision tree algorithms further enhances the predictive capabilities of the system. Specifically, diseases such as Diabetes, Malaria, Jaundice, Dengue, and Tuberculosis are targeted for prediction through the utilization of these machine learning techniques.

This methodological approach leverages the power of machine learning to contribute to the advancement of healthcare by facilitating timely disease identification and proactive patient management. The system aims to harness the potential of data-driven insights for the benefit of medical practitioners and the overall improvement of public health outcomes.

Keyword: Disease Prediction, Machine Learning, KNN classifier, Decision tree, Random Forest, Medical assistance.

सार:

मशीन लर्निंग सिस्टम का उपयोग करके रोग की भविष्यवाणी स्वास्थ्य देखभाल में एक अभूतपूर्व दृष्टिकोण का प्रतिनिधित्व करती है, जो उपयोगकर्ता द्वारा प्रदान किए गए लक्षणों के आधार पर बीमारियों की घटना का अनुमान लगाने के लिए उन्नत पूर्वानुमान मॉडलिंग तकनीकों का उपयोग करती है। सिस्टम विशिष्ट बीमारियों की संभावना की गणना करने के लिए के-निकटतम पड़ोसियों (केएनएन) क्लासिफायरियर, डिसीजन ट्री और रैंडम फ़ॉरेस्ट एल्गोरिदम का एक मजबूत संयोजन नियोजित करता है। इस नवोन्मेषी पद्धति का उद्देश्य उपयोगकर्ता-इनपुट लक्षणों को संसाधित करके और संभावित बीमारियों से जुड़े संभाव्यता आउटपुट उत्पन्न करके रोग का पता लगाने में क्रांति लाना और रोगी देखभाल को बढ़ाना है।

बायोमेडिकल और हेल्थकेयर डेटा की बढ़ती मात्रा के जवाब में, सिस्टम चिकित्सा जानकारी के सटीक विश्लेषण की महत्वपूर्ण आवश्यकता को संबोधित करता है। इसका प्राथमिक उद्देश्य रोग का शीघ्र पता लगाना है, जो रोगी के सर्वोत्तम परिणाम सुनिश्चित करने में एक महत्वपूर्ण कारक है। रैखिक प्रतिगमन और निर्णय वृक्ष एल्गोरिदम का एकीकरण प्रणाली की पूर्वानुमान क्षमताओं को और बढ़ाता है, जिससे रोग की भविष्यवाणी के लिए एक व्यापक और सूक्ष्म दृष्टिकोण प्रदान होता है।

मधुमेह, मलेरिया, पीलिया, डेंगू और तपेदिक जैसी प्रचलित बीमारियों को लक्षित करते हुए, सिस्टम सटीक और समय पर पूर्वानुमान देने के लिए मशीन लर्निंग तकनीकों की शक्ति का लाभ उठाता है। उपयोगकर्ता-इनपुट लक्षणों का विश्लेषण करके, मॉडल प्रत्येक संभावित बीमारी से जुड़ी संभावनाएं उत्पन्न करता है, जो चिकित्सा चिकित्सकों को सूचित निर्णय लेने में एक मूल्यवान उपकरण प्रदान करता है।

विशेषं रूप से, यह प्रणाली केवल भविष्यवाणी से परे है, क्योंकि यह अनुमानित बीमारी के आधार पर अनुरूप चिकित्सा सहायता भी प्रदान करती है। मॉडल का यह पहलू सक्रिय रोगी प्रबंधन में महत्वपूर्ण योगदान देता है, जिससे शीघ्र हस्तक्षेप और व्यक्तिगत देखभाल रणनीतियों की अनुमति मिलती है।

TABLE OF CONTENTS

TITLE **PAGE NO.**

LIST OF FIGURES

LIST OF TABLES

Chapter 1: Project Overview

1.1 Introduction to Machine Learning in Healthcare:

In recent years, the integration of machine learning techniques into healthcare systems [1, 2] has emerged as a transformative force, offering unprecedented opportunities for disease prediction and prevention. The conventional healthcare paradigm, reliant on manual analysis and historical data, faces limitations in terms of accuracy and efficiency. The burgeoning availability of healthcare data, coupled with advancements in computational capabilities, has fuelled a paradigm shift towards leveraging machine learning for disease prediction [3].

1.2 Role of Machine Learning in Disease Prediction:

Machine learning uses different methods like statistics, probabilities, and optimizations to learn from past data. It helps identify important patterns in large, unstructured and messy datasets [4]. These algorithms are used in various areas such as sorting text automatically, network intrusion detection [5], filtering out junk e-mail [6], detecting credit card fraud [7], understanding how customers make purchases, optimizing production manufacturing [8], and disease modelling [9, 10, 11]. Most of these uses rely on a type of machine learning called "supervised learning," [6, 7, 9] where the system learns from labelled data (data with known outcomes) to make predictions about new, unlabelled data. Early detection of diseases is a critical factor in improving patient outcomes and reducing healthcare costs. Machine learning algorithms, with their ability to discern intricate patterns within vast datasets, have demonstrated remarkable potential in enhancing the accuracy of disease prediction models [9, 11, 15].

1.3 Research Motivation and Objectives:

The motivation behind this project lies in the need to comprehensively understand the current state-of-the-art in disease prediction using machine learning, identify successful methodologies, and address existing challenges. As healthcare providers and researchers increasingly recognize the value of predictive analytics, it becomes imperative to explore the nuances of these techniques, their limitations [16], and potential avenues for improvement [17].

Chapter 2: METHODS

2.1 Overview:

This section provides a comprehensive overview of the process involved in creating the dataset, preparing the model, and predicting diseases. The initial step involves meticulous data collection from diverse sources. Subsequently, the collected data undergoes a thorough preprocessing phase, dividing it into cleaning and test datasets. Following data preparation, the training dataset is subjected to machine learning algorithms, specifically KNN [7], Decision tree, Random Forest [6], over multiple epochs to enhance the accuracy of prediction outcomes. Once the model achieves the predetermined target accuracy through these training cycles, it is deemed ready for testing.

The testing phase evaluates the model's performance using a completely new set of data that was not part of the training process. If the model demonstrates the desired accuracy with this fresh test data, it signifies that the proposed model is prepared for deployment, ensuring reliability and effectiveness in predicting diseases, then the proposed model is ready for deployment as shown in Fig 1.

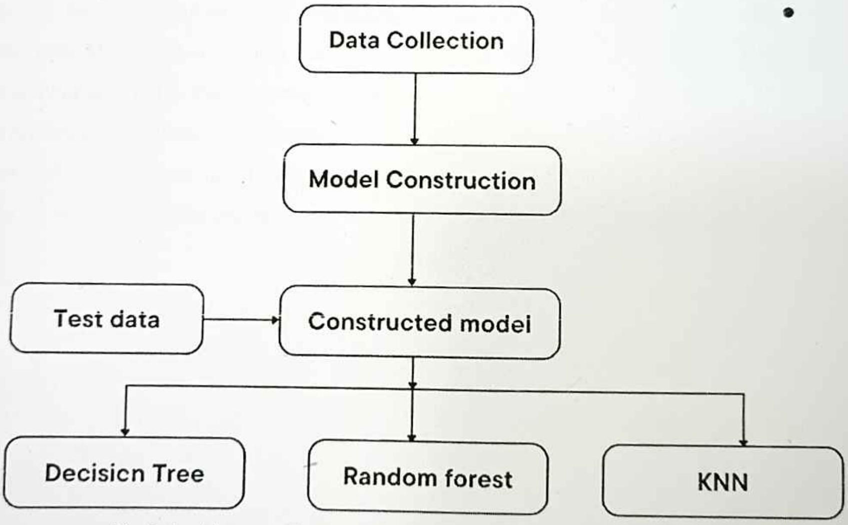


Fig 1: Architecture of proposed disease and risk prediction system.

2.2 Supervised machine learning algorithms:

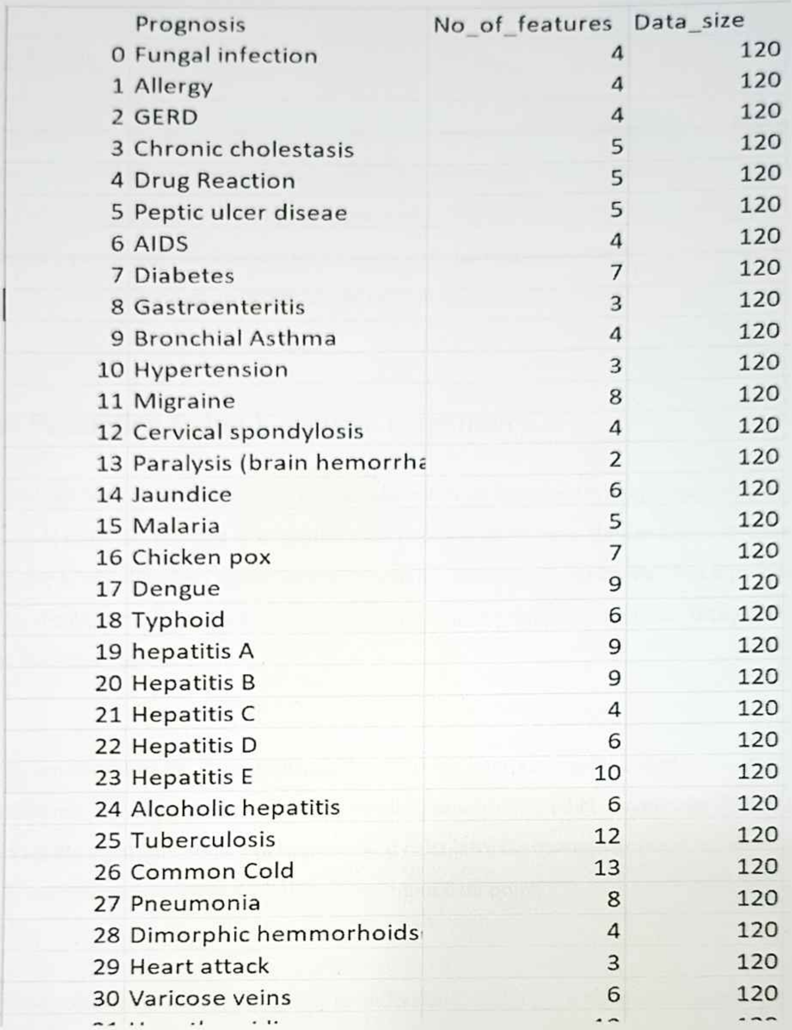
Machine learning algorithms are computational models that allow computers to understand patterns and forecast or make judgments based on data without the need for explicit programming. These algorithms form the foundation of modern artificial intelligence and are used in a wide range of applications, including image and speech recognition, natural language processing, recommendation systems, fraud detection, autonomous cars, healthcare system etc. Machine learning algorithms can be broadly categorized into three groups based on their purposes and how they're trained: supervised, unsupervised, and semi-supervised. These distinctions capture the varied ways in which machines are taught and contribute to the diverse landscape of machine learning applications.

Supervised learning uses a training set to teach models to yield the desired output. This training dataset includes inputs and correct outputs, which allow the model to learn over time. The algorithm measures its accuracy through the loss function, adjusting until the error has been sufficiently minimized. Supervised learning can be separated into two types of problems when data mining—classification and regression.

2.3 Data Collection:

data collection is the process of gathering data relevant to the ML project's goals and objectives. Collecting data for training the ML model is the basic step in the machine learning pipeline. The predictions made by ML systems can only be as good as the data on which they have been trained.

The disease prediction model utilizes authentic real-life data that includes structured data such as patient basic information including demographics, living habitat, lab test results and symptoms of the disease faced by the patient. The data set excludes the patient's personal details such as name, ID, and location so as to preserve their privacy.



	Prognosis	No_of_features	Data_size
0	Fungal infection	4	120
1	Allergy	4	120
2	GERD	4	120
3	Chronic cholestasis	5	120
4	Drug Reaction	5	120
5	Peptic ulcer diseae	5	120
6	AIDS	4	120
7	Diabetes	7	120
8	Gastroenteritis	3	120
9	Bronchial Asthma	4	120
10	Hypertension	3	120
11	Migraine	8	120
12	Cervical spondylosis	4	120
13	Paralysis (brain hemorrha	2	120
14	Jaundice	6	120
15	Malaria	5	120
16	Chicken pox	7	120
17	Dengue	9	120
18	Typhoid	6	120
19	hepatitis A	9	120
20	Hepatitis B	9	120
21	Hepatitis C	4	120
22	Hepatitis D	6	120
23	Hepatitis E	10	120
24	Alcoholic hepatitis	6	120
25	Tuberculosis	12	120
26	Common Cold	13	120
27	Pneumonia	8	120
28	Dimorphic hemmorhoids	4	120
29	Heart attack	3	120
30	Varicose veins	6	120

Fig 2. overview of a dataset used for the model

2.4 Preprocessing:

Data preprocessing is an essential phase in readying raw data for machine learning algorithms, serving to refine, organize, and enhance its quality. The process involves several key steps. Firstly, data cleaning addresses missing values and identifies and manages outliers that could distort analyses. Transformation operations include encoding categorical variables into numerical formats and scaling numerical features to ensure equal contribution. Feature engineering may involve creating or modifying features to optimize model performance. Data reduction techniques, such as dimensionality reduction and sampling, aim to enhance model efficiency.

2.5 Data splitting:

In machine learning, the process of splitting the dataset into training and testing sets is a fundamental step to evaluate the performance of a model on unseen data. This separation helps ensure that the model, once trained on the training set, can generalize well to new, previously unseen examples. The typical approach involves allocating a certain percentage of the data for training and the remaining portion for testing. Common splits include an 80-20 or 70-30 division, with the larger portion designated for training.

2.6 Disease Prediction Using K-nearest neighbour:

The K-Nearest Neighbour (KNN) algorithm is a popular machine learning technique used for classification and regression tasks. It relies on the idea that similar data points tend to have similar labels or values.During the training phase, the KNN algorithm stores the entire training dataset as a reference. When making predictions, it calculates the distance between the input data point and all the training examples, using a chosen distance metric such as Euclidean distance.

Next, the algorithm identifies the K nearest neighbours to the input data point based on their distances. In the case of classification, the algorithm assigns the most common class label among the K neighbours as the predicted label for the input data point. For regression, it calculates the average or weighted average of the target values of the K neighbours to predict the value for the input data point.

The KNN algorithm is straightforward and easy to understand, making it a popular choice in various domains. However, its performance can be affected by the choice of K and the distance metric, so careful parameter tuning is necessary for optimal results. Fig 3 shows an illustration of how the KNN works to classify a disease

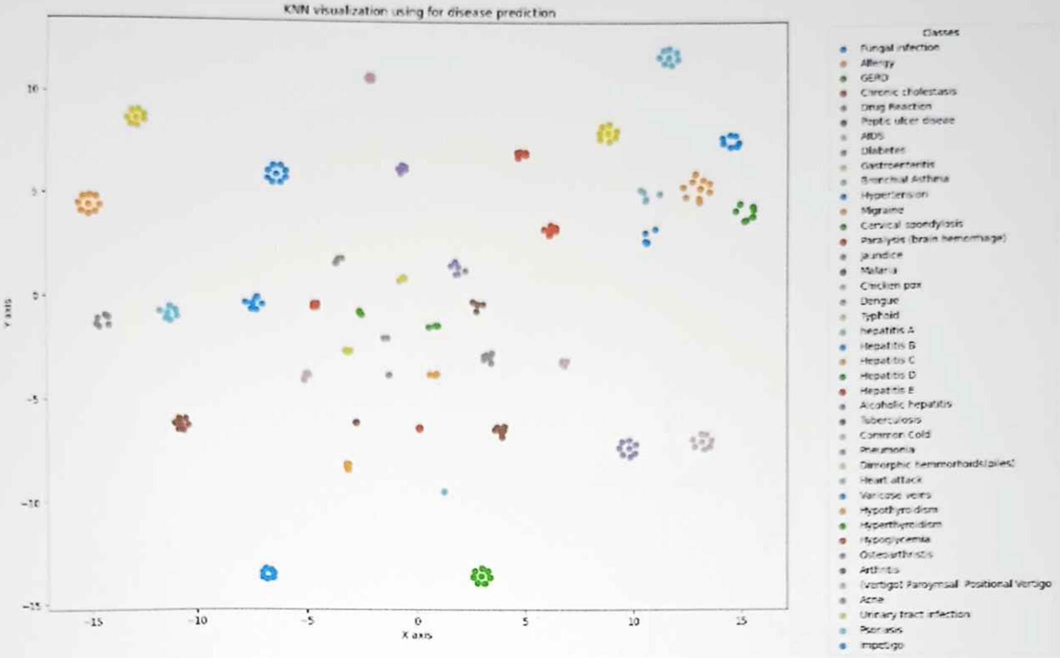


Fig 3: Illustration of how the KNN classifier works.

2.7 Disease Prediction Using Decision tree:

A decision tree is one of the most powerful tools of supervised learning algorithms used for both classification and regression tasks. It builds a flowchart-like tree structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. It is constructed by recursively splitting the training data into subsets based on the values of the attributes until a stopping criterion is met, such as the maximum depth of the tree or the minimum number of samples required to split a node.

During training, the Decision Tree algorithm selects the best attribute to split the data based on a metric such as entropy or Gini impurity, which measures the level of impurity or randomness in the subsets. The goal is to find the attribute that maximizes the information gain or the reduction in impurity after the split. Fig 4 shows an illustration of how the decision tree works.

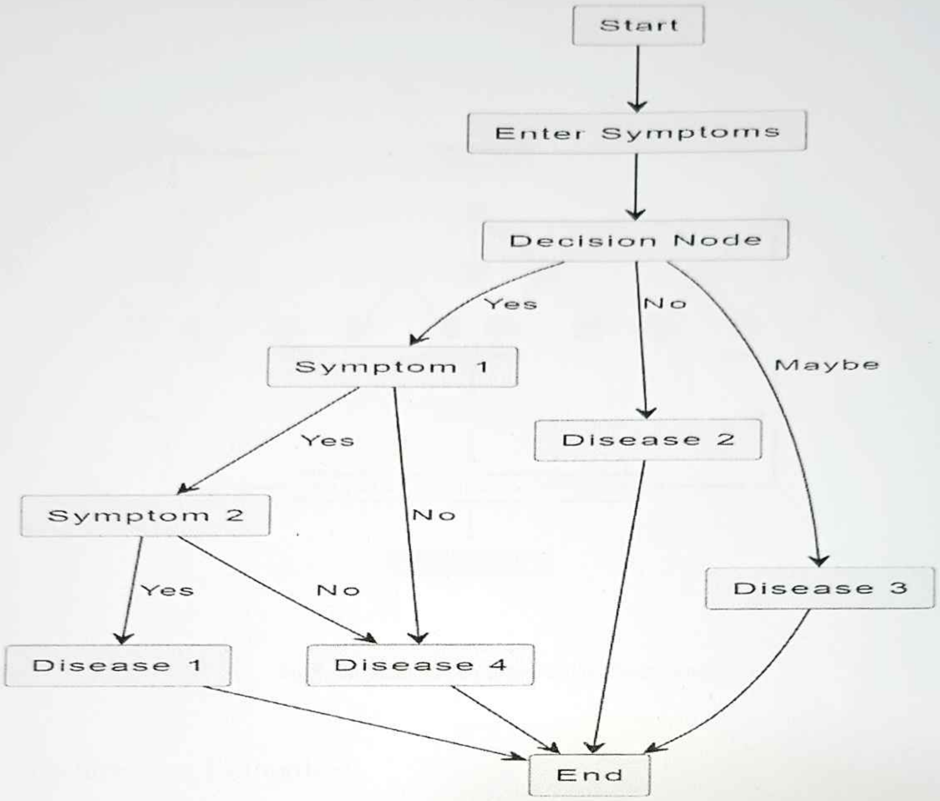


Fig 4: Illustration of how the decision tree works.

2.8 Disease Prediction Using Random Forest:

A random forest (RF) is an ensemble classifier and consisting of many DTs similar to the way a forest is a collection of many trees [20]. The Random Forest algorithm is like a smart system that combines the strengths of many decision trees. However, sometimes a single decision tree can be too specific and make predictions that work well on the data it was trained on but might not be accurate for new data. Instead of relying on just one decision tree, it creates a whole bunch of them. Each tree is like a small expert specializing in certain aspects of the data. When it's time to make a prediction, all these experts vote, and the majority opinion is taken as the final prediction. This teamwork helps reduce the risk of making predictions that are too tailored to the training data and might not work well for new cases. Moreover, Random Forest is good at telling us which factors (or symptoms in the case of disease prediction) are the most important in making predictions. It looks at all the decision trees and figures out which symptoms are consistently crucial across the board. This way, it not only gives accurate predictions but also helps us understand which symptoms are most linked to the disease we are trying to predict. Fig 5 shows an illustration of the RF algorithm.

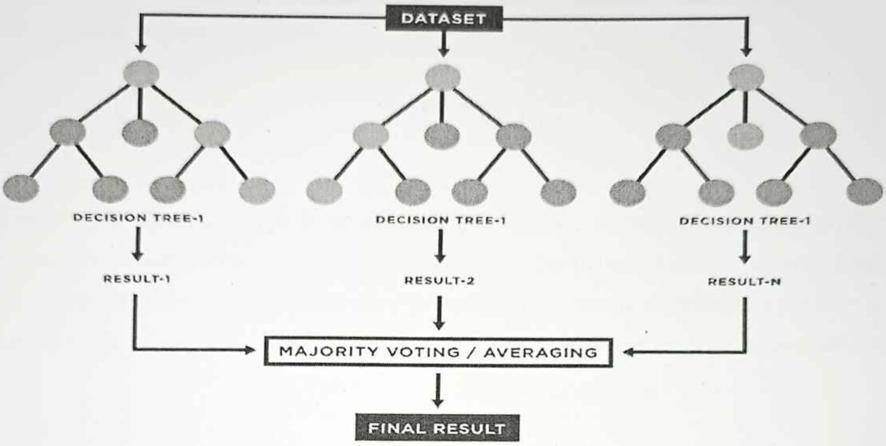


Fig 5: Illustration of how the Random Forest works.

2.9 Performance Evaluation:

To assess our disease prediction model, we use four key metrics. The confusion matrix helps us understand how well the model performs. True Positives (TP) represent when the model correctly predicts someone with a chronic disease. True Negatives (TN) occur when it accurately identifies individuals without diseases. False Positives (FP) happen when the model mistakenly predicts a healthy person as having a disease, and False Negatives (FN) occur when it incorrectly predicts someone with a chronic disease as being healthy. The following is the description of the four performance evaluation parameters.

Accuracy. The classification accuracy is described as the ratio of correct predicted values to the total predicted values and is depicted mathematically as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} * 100 \qquad (1)$$

Precision. The precision or positive predictive value (PPV) is described as the ratio of correct prediction to the total correct values including the true and false predictions and is depicted mathematically as follows:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Recall. The recall or sensitivity or true positive rate (TPR) is described as the ratio of correct predicted values to the sum of correct positive predictions and the incorrect negative predicted values and is depicted mathematically as follows:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

F1-Score. The F-measure (Fβ) is described as the weighted average of the values obtained from the calculation of precision and recall parameters. Whenever the distribution of class is not even, then the value of F1 – Score is highly important than the accuracy value. And whenever the values of false positives and negatives are dissimilar, the value of F1 – Score is highly suitable. The F1 – Score is depicted mathematically as follows:

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4}$$

Chapter 3: Final Analysis and Design

3.1 Accuracy comparison

Model	Accuracy
Decision tree	97.63
KNN	97.19
Random Forest	98.72

Table 1: Shows the accuracy comparison model trained with different ML algorithms

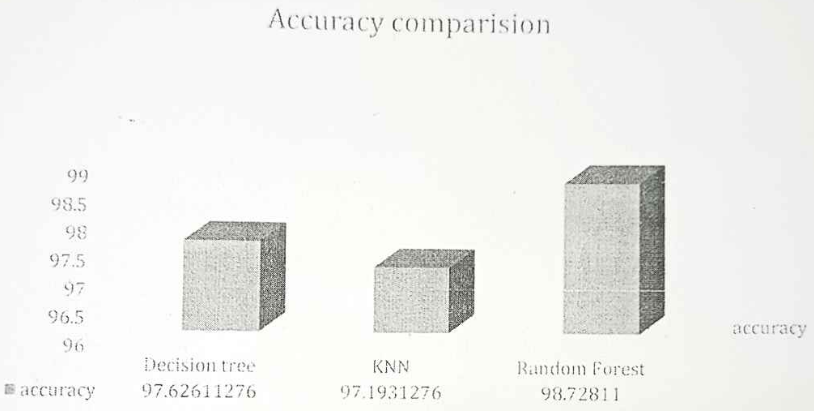


Fig 6: Shows the accuracy comparison model trained with different ML algorithms

3.2 Code [Model train]

The provided code is designed to train a machine learning model for disease prediction using various algorithms. It likely involves importing necessary libraries, loading a dataset, and applying different machine learning algorithms, such as decision trees, KNN, and Random Forest to train predictive models. The goal is to identify patterns and relationships within the data that can accurately predict the occurrence of diseases. The specifics of the code depend on its details, which are not provided.

```
In [ ]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
```

```
In [ ]: # data=pd.read_csv('trainingdata.csv')
        data=pd.read_csv('NEWTRAIN.csv')
```

```
In [ ]: data.head()
```

Out[ ]:

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue
0	1	1	1	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0	0	0
4	1	1	1	0	0	0	0	0	0	0

5 rows × 519 columns

```
In [ ]: data.columns

Out[ ]: Index(['itching', 'skin_rash', 'nodal_skin_eruptions', 'continuous_sneezing',
               'shivering', 'chills', 'joint_pain', 'stomach_pain', 'acidity',
               'ulcers_on_tongue',
               ...
               'prodrome', 'hypoproteinemia', 'alcohol binge episode', 'abdomen acute',
               'air fluid level', 'catching breath', 'large-for-dates fetus',
               'immobile', 'homicidal thoughts', 'prognosis'],
              dtype='object', length=519)
```

```
In [ ]: import seaborn as sns
        plt.figure(figsize=(10,10))
        sns.barplot(y='prognosis',x='abdominal_pain',data=data)
        plt.savefig('abdominal_pain.pdf')
```

```
In [ ]: d=data[data['prognosis']=='diabetes']
```

```
In [ ]: len(data.prognosis.unique())

Out[ ]: 174
```

```
In [ ]: data.prognosis.unique()[0]

Out[ ]: 'Fungal infection'
```

```
In [ ]: d.drop(['prognosis'],axis=1).replace(0,np.nan).dropna(axis=1,how="all").columns

Out[ ]: Index(['vomiting', 'nausea', 'polyuria', 'pain chest', 'shortness of breath',
               'asthenia', 'vertigo', 'sweat', 'polydypsia', 'orthopnea', 'rale',
               'unresponsiveness', 'mental status changes', 'labored breathing'],
              dtype='object')
```

```
In [ ]: new_data=pd.DataFrame(columns=['prognosis','No_of_symptoms','Data_size'])
```

```
In [ ]: new_data
```

Out[ ]:

prognosis	No_of_symptoms	Data_size

```
In [ ]: new_data
```

Out[ ]:

prognosis	No_of_symptoms	Data_size

```
In [ ]: new_data.head()
```

Out[ ]:

prognosis	No_of_symptoms	Data_size

```
In [ ]: new_data
```

Out[ ]:

	prognosis	No_of_symptoms	Data_size

```
In [ ]: new_data.to_csv('new_data.csv')
```

```
In [ ]: new_data.head()
```

Out[ ]:

	prognosis	No_of_symptoms	Data_size

```
In [ ]: d.skin_rash.value_counts()
```

```
Out[ ]: skin_rash
        0    1
        Name: count, dtype: int64
```

```
In [ ]: data.dtypes
```

```
Out[ ]: itching                    int64
        skin_rash                  int64
        nodal_skin_eruptions       int64
        continuous_sneezing        int64
        shivering                  int64
                                   ...
        catching breath            int64
        large-for-dates fetus      int64
        immobile                   int64
        homicidal thoughts         int64
        prognosis                 object
        Length: 519, dtype: object
```

```
In [ ]: len(data['prognosis'].value_counts())
```

```
Out[ ]: 174
```

```
In [ ]: import seaborn as sns
```

```
In [ ]: a=data['prognosis'].drop_duplicates()
```

```
In [ ]: from sklearn import preprocessing
        label_encoder1 = preprocessing.LabelEncoder()
        data['prognosis']= label_encoder1.fit_transform(data['prognosis'])
```

```
In [ ]: from joblib import parallel, delayed
        import joblib

        joblib.dump(label_encoder1,'encode.pkl')
```

```
Out[ ]: ['encode.pkl']
```

```
In [ ]: b=data['prognosis'].drop_duplicates()
```

```
In [ ]: result=pd.concat([a,b],axis=1)
```

```
In [ ]: result.to_csv('result_label_en.csv')
```

```
In [ ]: data.head()
```

Out[ ]:

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue
0	1	1	1	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0	0	0
4	1	1	1	0	0	0	0	0	0	0

5 rows × 519 columns

```
In [ ]: # b['prognosis']
In [ ]: # s = ''
        # with open ("name.csv") as annotate:
        #     for col in annotate:
        #         name = col.lower().split(",")
        #         s += name[0] + ",'"
        # s = s[:-1] # Remove last comma
        # print(s)
```

```
In [ ]: # label_encoder.inverse_transform(data['prognosis'])
```

```
In [ ]: data.columns
```

```
Out[ ]: Index(['itching', 'skin_rash', 'nodal_skin_eruptions', 'continuous_sneezing',
               'shivering', 'chills', 'joint_pain', 'stomach_pain', 'acidity',
               'ulcers_on_tongue',
               ...
               'prodrome', 'hypoproteinemia', 'alcohol binge episode', 'abdomen acute',
               'air fluid level', 'catching breath', 'large-for-dates fetus',
               'immobile', 'homicidal thoughts', 'prognosis'],
              dtype='object', length=519)
```

```
In [ ]: # sns.scatterplot(x='prognosis',y='stomach_pain',data=data)
```

```
In [ ]: # data.drop(['Unnamed: 133'],axis=1,inplace=True)
```

```
In [ ]: x=data.drop(['prognosis'],axis=1)
        y=data['prognosis']
```

```
In [ ]: from sklearn.model_selection import train_test_split
        x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

```
In [ ]: from sklearn.neighbors import KNeighborsClassifier
        from sklearn.metrics import accuracy_score
```

```
In [ ]: final_scores = []
        for i in range(1,30,2):
            knn = KNeighborsClassifier(n_neighbors = i)
            knn.fit(x_train, y_train)
            pred = knn.predict(x_test)
            acc = accuracy_score(y_test, pred, normalize=True) * float(100)
            final_scores.append(acc)
            print('\n CV accuracy for k=%d is %d'%(i,acc))
```

```
 CV accuracy for k=1 is 97

 CV accuracy for k=3 is 97

 CV accuracy for k=5 is 97

 CV accuracy for k=7 is 97

 CV accuracy for k=9 is 97

 CV accuracy for k=11 is 97

 CV accuracy for k=13 is 97

 CV accuracy for k=15 is 97

 CV accuracy for k=17 is 97

 CV accuracy for k=19 is 97

 CV accuracy for k=21 is 97

 CV accuracy for k=23 is 97

 CV accuracy for k=25 is 97

 CV accuracy for k=27 is 97

 CV accuracy for k=29 is 97
```

```
In [ ]: optimal_k = final_scores.index(max(final_scores))
        print(optimal_k)
```

```
0
```

```
In [ ]: knn = KNeighborsClassifier(n_neighbors = 5)
        knn.fit(x_train, y_train)
```

```
Out[ ]: ▾ KNeighborsClassifier
        KNeighborsClassifier()
```

```
In [ ]: y_pred_knn=knn.predict(x_test)
        y_pred_knn
```

```
Out[ ]: array([18, 26, 13, ...,  7, 18, 16])
```

```
In [ ]: accuracy_score(y_test,y_pred_knn)
```

```
Out[ ]: 0.9762611275964391
```

```
In [ ]: # # Confusion Matrix heatmap
        # from sklearn import metrics
        # cm = metrics.confusion_matrix(y_test, y_pred)
        # cm

        # plt.figure(figsize=(100,100))
        # sns.heatmap(cm,annot=True,fmt='.0f')
        # plt.xlabel('Predicted')
        # plt.ylabel('Truth')
```

```
In [ ]: from sklearn.metrics import confusion_matrix,classification_report,ConfusionMatrixDisplay
        print(classification_report(y_test,y_pred_knn))
        confusion_matrix(y_test,y_pred_knn)
```

```
In [ ]: from sklearn.tree import DecisionTreeClassifier
        DTC=DecisionTreeClassifier()
```

```
In [ ]: DTC.fit(x_train,y_train)
```

```
Out[ ]: ▾ DecisionTreeClassifier
        DecisionTreeClassifier()
```

```
In [ ]: y_pred_dtc=DTC.predict(x_test)
        y_pred_dtc
```

```
Out[ ]: array([18, 26, 13, ...,  7, 18, 16])
```

```
In [ ]: y_pred_1=DTC.predict(x_test[y_test==27])
        y_pred_1
```

```
Out[ ]: array([27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
               27, 27, 27, 27, 27, 27])
```

```
In [ ]: from sklearn.metrics import accuracy_score
        accuracy_score(y_test,y_pred_dtc)*100
```

```
Out[ ]: 97.62611275964392
```

```
In [ ]: # from sklearn.metrics import accuracy_score
        # accuracy_score(y_test[y_test==30],y_pred_1)*100
```

```
In [ ]: # from sklearn.metrics import confusion_matrix,classification_report,ConfusionMatrixDisplay
        # con=confusion_matrix(y_test,y_pred)
        # ConfusionMatrixDisplay(con).plot()
        # plt.show()
```

```
In [ ]: from sklearn.metrics import confusion_matrix,classification_report,ConfusionMatrixDisplay
        print(classification_report(y_test,y_pred_dtc))
        confusion_matrix(y_test,y_pred_dtc)
```

```
In [ ]: print(y_test,y_pred_dtc)

        4026    18
        4379    26
        3779    13
        39      10
        867     15
                ..
        2952    16
        3953    12
        2838     7
        4231    18
        1642    16
        Name: prognosis, Length: 1011, dtype: int32 [18 26 13 ...  7 18 16]
```

```
In [ ]: y_test=np.array(y_test)
        y_pred=np.array(y_pred_dtc)
        y_pred[0]
        y_test[0]

Out[ ]: 18
```

```
In [ ]: y_test

Out[ ]: array([18, 26, 13, ...,  7, 18, 16])
```

```
In [ ]: correct=0
        incorrect=0
        for i in range(len(y_test)):
            if (y_pred[i] == y_test[i]):
                # print("CORRECT: ",y_pred[i],',',y_test[i])
                correct+=1
            else:
                # print("INCORRECT:",y_pred[i],',',y_test[i])
                incorrect+=1

        print("Correct:",correct)
        print("Incorrect:",incorrect)

Correct: 987
Incorrect: 24
```

random forest

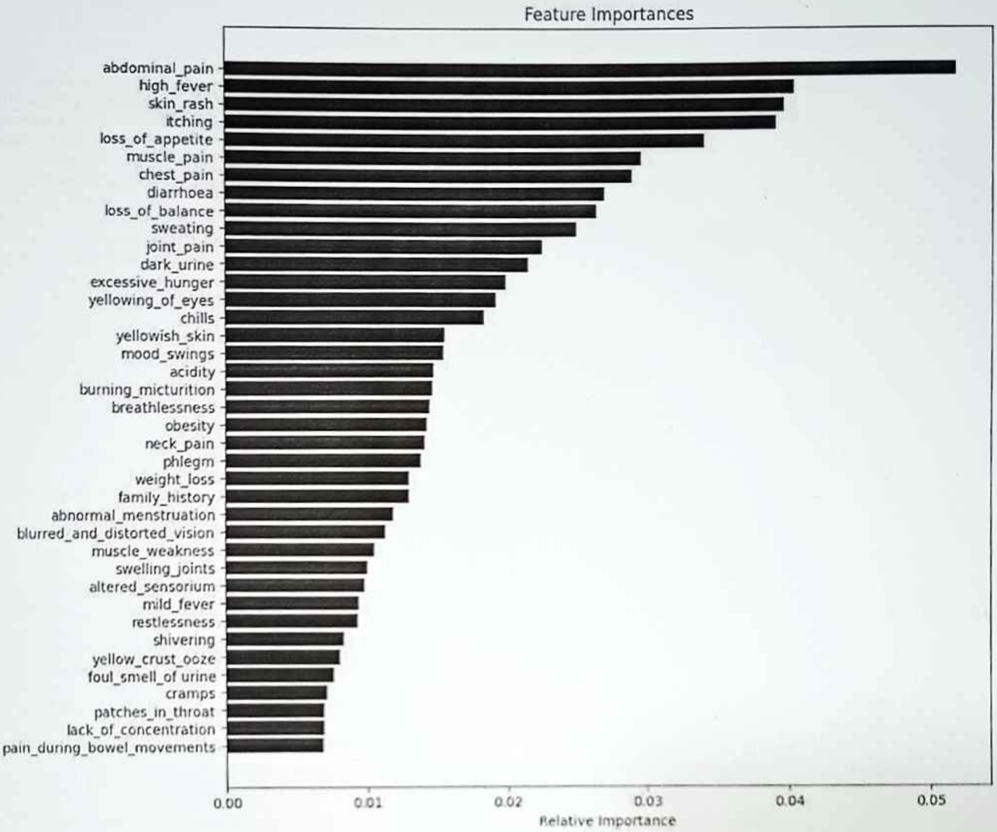
```
In [ ]: from sklearn.ensemble import RandomForestClassifier
        classifier= RandomForestClassifier(n_estimators= 10, criterion="entropy")
        classifier.fit(x_train, y_train)

Out[ ]: ▾                 RandomForestClassifier
        RandomForestClassifier(criterion='entropy', n_estimators=10)
```

```
In [ ]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                               max_depth=None, max_features='auto', max_leaf_nodes=None,
                               min_impurity_decrease=0.0,
                               min_samples_leaf=1, min_samples_split=2,
                               min_weight_fraction_leaf=0.0, n_estimators=10,
                               n_jobs=None, oob_score=False, random_state=None,
                               verbose=0, warm_start=False)

Out[ ]: ▾                   RandomForestClassifier
        RandomForestClassifier(criterion='entropy', max_features='auto',
                               n_estimators=10)
```

```
In [ ]: y_pred_rf= classifier.predict(x_test)
```

```
In [ ]: accuracy_score(y_test,y_pred_rf)*100

Out[ ]: 97.62611275964392
```

```
In [ ]: features = data.columns
        importances = classifier.feature_importances_
        indices = np.argsort(importances)[-39:]  # top 40 features
        plt.figure(figsize=(10,10))
        plt.title('Feature Importances')
        plt.barh(range(len(indices)), importances[indices], color='b', align='center')
        plt.yticks(range(len(indices)), [features[i] for i in indices])
        plt.xlabel('Relative Importance')
        plt.show()
```



In [ ]:

```
a=[]
for i in range(len(indices)):
    print(features[indices[i]])
    a.append(features[indices[i]])
```

```
pain_during_bowel_movements
lack_of_concentration
patches_in_throat
cramps
foul_smell_of urine
yellow_crust_ooze
shivering
restlessness
mild_fever
altered_sensorium
swelling_joints
muscle_weakness
blurred_and_distorted_vision
abnormal_menstruation
family_history
weight_loss
phlegm
neck_pain
obesity
breathlessness
burning_micturition
acidity
mood_swings
yellowish_skin
chills
yellowing_of_eyes
excessive_hunger
dark_urine
joint_pain
sweating
loss_of_balance
diarrhoea
chest_pain
muscle_pain
loss_of_appetite
itching
skin_rash
high_fever
abdominal_pain
```

```
In [ ]: a
```

```
Out[ ]: ['pain_during_bowel_movements',
         'lack_of_concentration',
         'patches_in_throat',
         'cramps',
         'foul_smell_of urine',
         'yellow_crust_ooze',
         'shivering',
         'restlessness',
         'mild_fever',
         'altered_sensorium',
         'swelling_joints',
         'muscle_weakness',
         'blurred_and_distorted_vision',
         'abnormal_menstruation',
         'family_history',
         'weight_loss',
         'phlegm',
         'neck_pain',
         'obesity',
         'breathlessness',
         'burning_micturition',
         'acidity',
         'mood_swings',
         'yellowish_skin',
         'chills',
         'yellowing_of_eyes',
         'excessive_hunger',
         'dark_urine',
         'joint_pain',
         'sweating',
         'loss_of_balance',
         'diarrhoea',
         'chest_pain',
         'muscle_pain',
         'loss_of_appetite',
         'itching',
         'skin_rash',
         'high_fever',
         'abdominal_pain']
```

```
In [ ]: data_new=data[['receiving_blood_transfusion',
         'depression',
         'extra_marital_contacts',
         'inflammatory_nails',
       'burning_micturition',
       'fluid_overload.1',
       'mood_swings',
       'neck_pain',
       'acidity',
       'swelling_joints',
       'stomach_pain',
       'swelled_lymph_nodes',
       'muscle_weakness',
       'painful_walking',
       'back_pain',
       'excessive_hunger',
       'stiff_neck',
       'sweating',
       'blurred_and_distorted_vision',
       'irritability',
       'family_history',
       'joint_pain',
       'mild_fever',
       'diarrhoea',
       'dark_urine',
       'itching',
       'phlegm',
       'muscle_pain',
       'high_fever',
       'breathlessness',
       'weight_loss',
       'abdominal_pain',
       'yellowing_of_eyes',
       'loss_of_appetite',
       'loss_of_balance',
       'chest_pain',
       'yellowish_skin',
       'chills',
       'skin_rash','prognosis']]
```

```
In [ ]: data_new.head()
```

Out[ ]:

	receiving_blood_transfusion	depression	extra_marital_contacts	inflammatory_nails	burning_micturition	fluid_overload.1	mood_sw
0	0	0	0	0	0	0	
1	0	0	0	0	0	0	
2	0	0	0	0	0	0	
3	0	0	0	0	0	0	
4	0	0	0	0	0	0	

5 rows × 40 columns

```
In [ ]: x1=data_new.drop(['prognosis'],axis=1)
        y1=data_new['prognosis']
```

```
In [ ]: from sklearn.model_selection import train_test_split
        x_train1,x_test1,y_train1,y_test1=train_test_split(x1,y1,test_size=0.2,random_state=0)
```

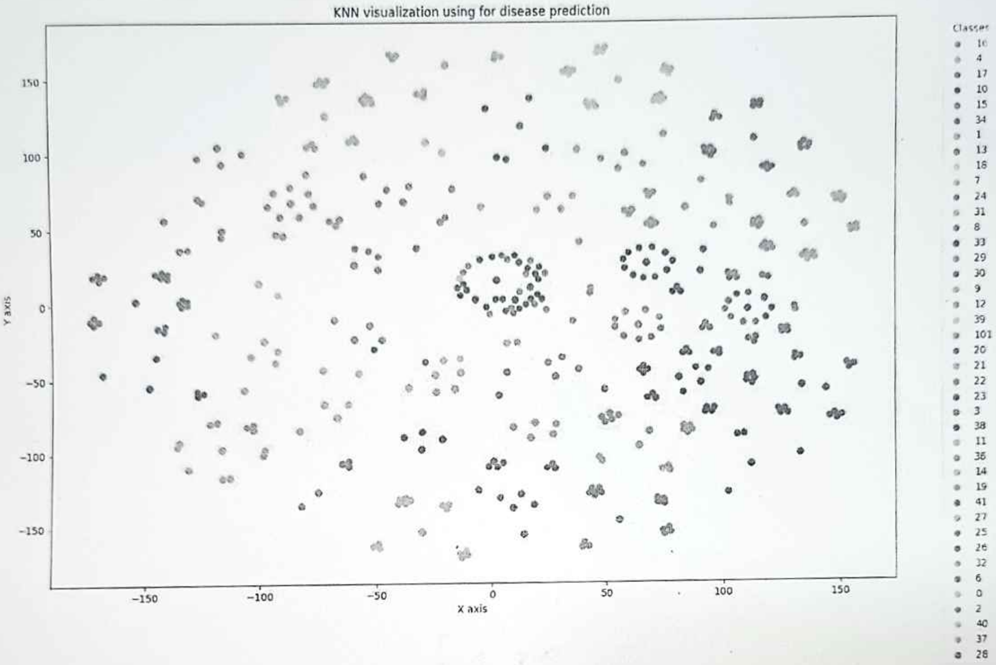
```
In [ ]: from sklearn.neighbors import KNeighborsClassifier
        from sklearn.metrics import accuracy_score
```

```
In [ ]: final_scores = []
        for i in range(1,30,2):
            knn = KNeighborsClassifier(n_neighbors = i)
            knn.fit(x_train1, y_train1)
            pred = knn.predict(x_test1)
            acc = accuracy_score(y_test1, pred, normalize=True) * float(100)
            final_scores.append(acc)
            print('\n CV accuracy for k=%d is %d'%(i,acc))
```

```
In [ ]: lab=data_new['prognosis'].unique()
```

```
In [ ]: from sklearn.neighbors import KNeighborsClassifier
        from sklearn.metrics import accuracy_score
        from matplotlib.colors import ListedColormap
        from sklearn.decomposition import PCA
        from sklearn.manifold import TSNE

        # Apply t-SNE for dimensionality reduction and visualization
        tsne = TSNE(n_components=2, perplexity=30, random_state=42)
        X_train_tsne = tsne.fit_transform(x_train1)

        # Visualize the t-SNE plot
```

```
plt.figure(figsize=(15, 10))
for i in range(41):
    indices = (y_train1 == i)
    plt.scatter(X_train_tsne[indices, 0], X_train_tsne[indices, 1], label=lab[i])

    # Annotate some points with class names for better visualization
    # if i % 5 == 0:
    #     for j in range(sum(indices)):
    #         plt.annotate(str(i), (X_train_tsne[indices, 0][j], X_train_tsne[indices, 1][j]))

# Add labels and Legend
plt.title("KNN visualization using for disease prediction")
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.legend(title="Classes", bbox_to_anchor=(1.05, 1), loc='upper left')

plt.show()
```



In [ ]:
```
data['prognosis']= label_encoder1.inverse_transform(data['prognosis'])
```

In [ ]:
```
data['prognosis']
```

Out[ ]:
```
0        Fungal infection
1        Fungal infection
2        Fungal infection
3        Fungal infection
4        Fungal infection
              ...
5048   tachycardia sinus
5049               ileus
5050            adhesion
5051            delusion
5052        affect labile
Name: prognosis, Length: 5053, dtype: object
```

In [ ]:
```
d=pd.read_csv('test.csv')
```

In [ ]:
```
d.isnull().sum()
```

```
Out[ ]: itching                   0
        skin_rash                 0
        nodal_skin_eruptions      0
        continuous_sneezing       0
        shivering                 0
                                 ..
        air fluid level           0
        catching breath           0
        large-for-dates fetus     0
        immobile                  0
        homicidal thoughts        0
        Length: 518, dtype: int64
```

```
In [ ]: y_pred1=DTC.predict(d)
        y_pred1
```

```
Out[ ]: array([16, 16, 16, 16, 16, 16,  4, 17, 10, 15, 34,  1, 13, 18,  7, 24, 31])
```

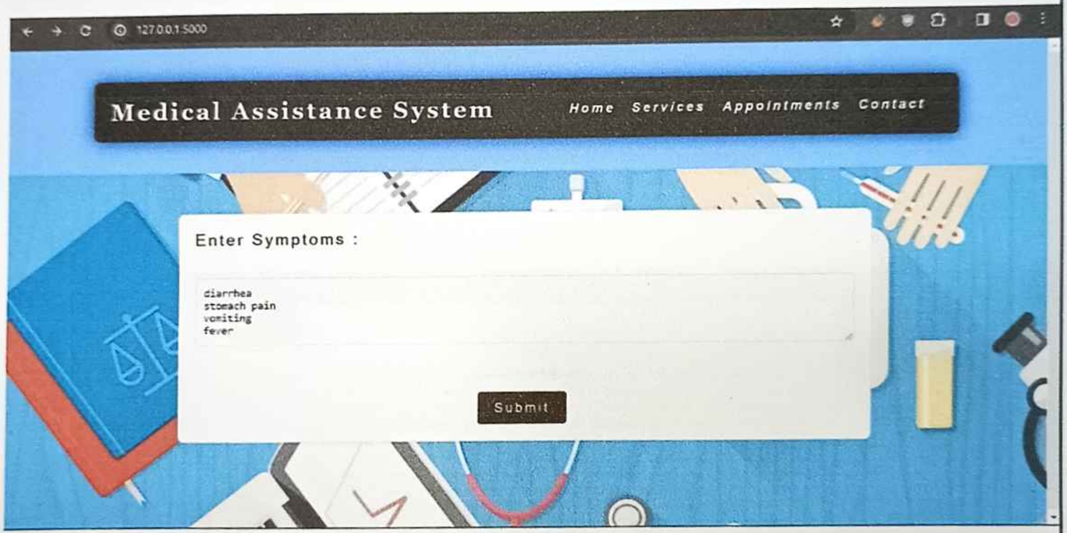
```
In [ ]: from joblib import parallel, delayed
        import joblib

        joblib.dump(DTC,'model.pkl')

        DTC_model=joblib.load('model.pkl')

        DTC_model.predict(d)
```

```
Out[ ]: array([16, 16, 16, 16, 16, 16,  4, 17, 10, 15, 34,  1, 13, 18,  7, 24, 31])
```

3.3 Code [Python program]

```
from fuzzywuzzy import fuzz
import numpy as np
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
from joblib import parallel, delayed
import joblib
from flask import Flask, render_template, url_for, request

app = Flask(__name__)

@app.route('/')
@app.route('/home')
def home():
    return render_template("index.html")



@app.route('/result',methods=['POST', 'GET'])
def result():

    output = request.form.to_dict()
    if "key" in output and isinstance(output["key"], str):
        output["key"] = output["key"].replace('\r', '')
    # print(output)
    with open('input.txt', 'w') as file:
        file.write(output["key"])
        file.close()

    def read(data):
        column_names=data.columns
        with open(r'input.txt', 'r') as fp:
        # read all lines using readline()
            lines = fp.readlines()
            non_blank_lines = [line.strip() for line in lines if line.strip()]
            for row in lines:
            # check if string present on a current line
                for column_name in column_names:
                    word = column_name
                    # if row.find(word) != -1:
                    if fuzz.ratio(word, row) >= 80:
                        # print("Word matched :", fuzz.ratio(word, row) )
                        data[column_name]=1
                        # print('selected column is : ',column_name)
    data=pd.read_csv('input.csv')
    read(data)
    DTC_model=joblib.load('model.pkl')
    pred=DTC_model.predict(data)

    from sklearn import preprocessing
    label_en=joblib.load('label_en.pkl')
    a=label_en.inverse_transform(pred)
    print("Disease name a is ",a)
    if(a):
        disease_name1 = a
    medical_data=pd.read_csv('medical_assit.csv')
    disease_names=medical_data.name
    found=0
    for disease_name in disease_names:
        word=disease_name
        if fuzz.ratio(word, a) >= 70:
            # print("Word matched :", fuzz.ratio(word, a) )
            # print('Predicted disease is : ',disease_name)
            if(disease_name!=None):
                found=1

            name="Disease : "+ (str(disease_name1).strip("[]'")).title()

            overview =
medical_data[medical_data.name==disease_name].overview.values[0].strip("'")
            causes
=  medical_data[medical_data.name==disease_name].causes.values[0].strip("'")
            treatment =
(str(medical_data[medical_data.name==disease_name].treatment.values[0]).strip("'"))
.replace('\n', '')

            medication =
medical_data[medical_data.name==disease_name].medication.values[0].strip("'")
            home_remedies =
medical_data[medical_data.name==disease_name].home_remedies.values[0].strip("'")
            NoFound = None

    if(found==0):
        name= None
        overview = None
        causes = None
        treatment = None
        medication = None
        home_remedies = None
        NoFound = " "
    name1 = str(a).strip("[]'").title()

    return render_template('index.html', name = name, overview = overview, causes =
causes, treatment = treatment, medication = medication,  home_remedies =
home_remedies, NoFound = NoFound, name1 = name1)

if __name__ == "__main__":
    app.run(debug=True)
```

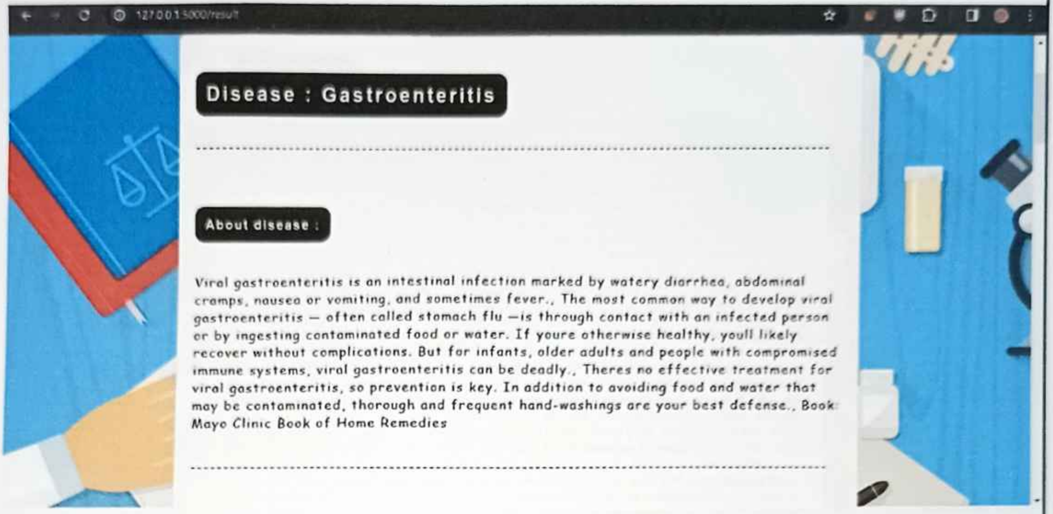
3.4 Results



Enter Symptoms :

diarrhea
stomach pain
vomiting
fever

Fig 7: Disease prediction symptoms entry form

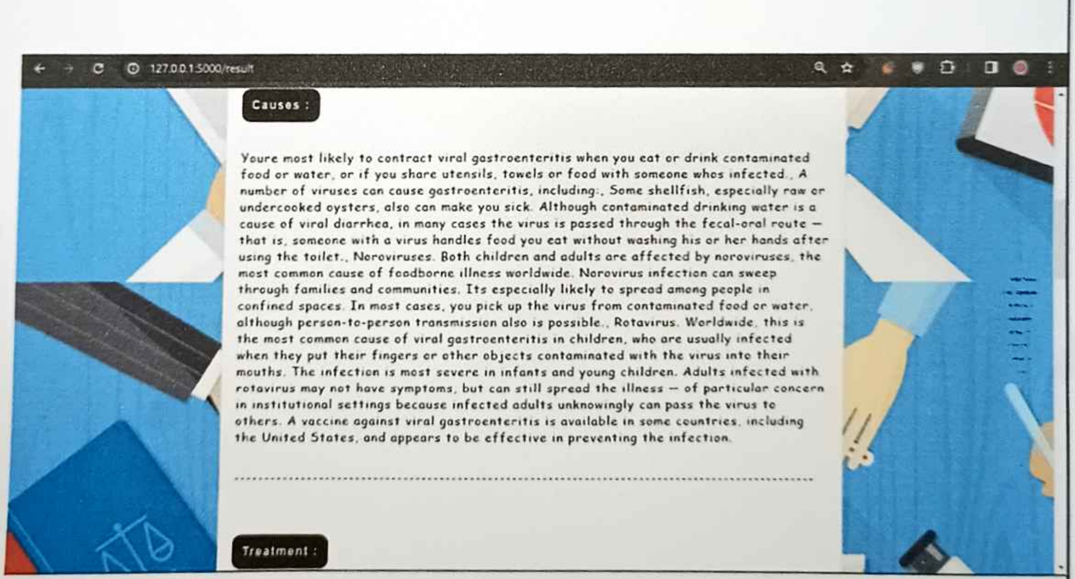


Disease : Gastroenteritis

About disease :

Viral gastroenteritis is an intestinal infection marked by watery diarrhea, abdominal cramps, nausea or vomiting, and sometimes fever., The most common way to develop viral gastroenteritis — often called stomach flu —is through contact with an infected person or by ingesting contaminated food or water. If youre otherwise healthy, youll likely recover without complications. But for infants, older adults and people with compromised immune systems, viral gastroenteritis can be deadly., Theres no effective treatment for viral gastroenteritis, so prevention is key. In addition to avoiding food and water that may be contaminated, thorough and frequent hand-washings are your best defense., Book: Mayo Clinic Book of Home Remedies

Fig 8: Disease predicted by the model

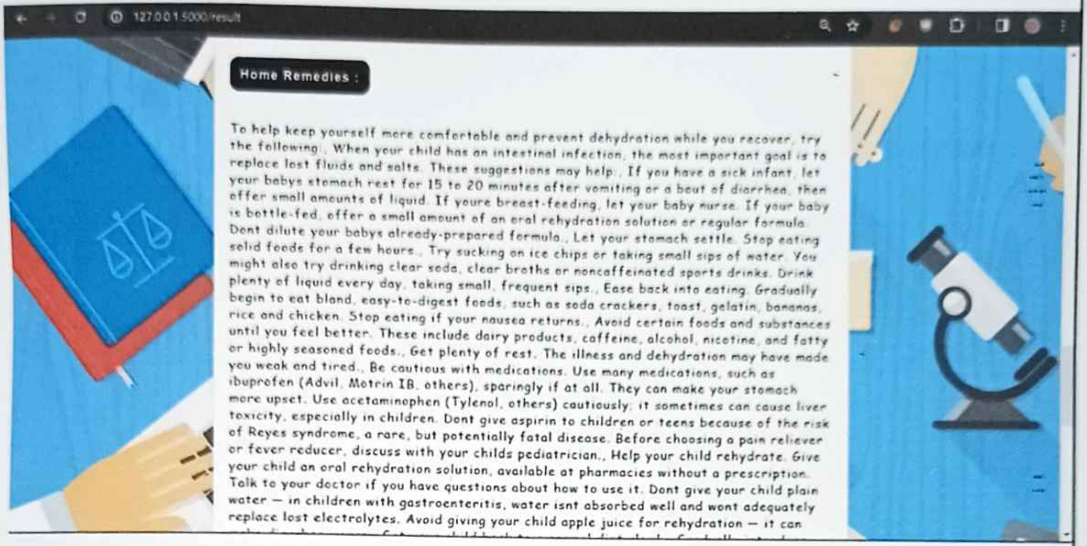


Causes :

Youre most likely to contract viral gastroenteritis when you eat or drink contaminated food or water, or if you share utensils, towels or food with someone whos infected., A number of viruses can cause gastroenteritis, including:, Some shellfish, especially raw or undercooked oysters, also can make you sick. Although contaminated drinking water is a cause of viral diarrhea, in many cases the virus is passed through the fecal-oral route — that is, someone with a virus handles food you eat without washing his or her hands after using the toilet., Noroviruses. Both children and adults are affected by noroviruses, the most common cause of foodborne illness worldwide. Norovirus infection can sweep through families and communities. Its especially likely to spread among people in confined spaces. In most cases, you pick up the virus from contaminated food or water, although person-to-person transmission also is possible., Rotavirus. Worldwide, this is the most common cause of viral gastroenteritis in children, who are usually infected when they put their fingers or other objects contaminated with the virus into their mouths. The infection is most severe in infants and young children. Adults infected with rotavirus may not have symptoms, but can still spread the illness — of particular concern in institutional settings because infected adults unknowingly can pass the virus to others. A vaccine against viral gastroenteritis is available in some countries, including the United States, and appears to be effective in preventing the infection.

Treatment :

Fig 9: shows the causes and treatment of disease



To help keep yourself more comfortable and prevent dehydration while you recover, try the following:, When your child has an intestinal infection, the most important goal is to replace lost fluids and salts. These suggestions may help:, If you have a sick infant, let your babys stomach rest for 15 to 20 minutes after vomiting or a bout of diarrhea, then offer small amounts of liquid. If youre breast-feeding, let your baby nurse. If your baby is bottle-fed, offer a small amount of an oral rehydration solution or regular formula. Dont dilute your babys already-prepared formula., Let your stomach settle. Stop eating solid foods for a few hours., Try sucking on ice chips or taking small sips of water. You might also try drinking clear soda, clear broths or noncaffeinated sports drinks. Drink plenty of liquid every day, taking small, frequent sips., Ease back into eating. Gradually begin to eat bland, easy-to-digest foods, such as soda crackers, toast, gelatin, bananas, rice and chicken. Stop eating if your nausea returns., Avoid certain foods and substances until you feel better. These include dairy products, caffeine, alcohol, nicotine, and fatty or highly seasoned foods., Get plenty of rest. The illness and dehydration may have made you weak and tired., Be cautious with medications. Use many medications, such as ibuprofen (Advil, Motrin IB, others), sparingly if at all. They can make your stomach more upset. Use acetaminophen (Tylenol, others) cautiously; it sometimes can cause liver toxicity, especially in children. Dont give aspirin to children or teens because of the risk of Reyes syndrome, a rare, but potentially fatal disease. Before choosing a pain reliever or fever reducer, discuss with your childs pediatrician., Help your child rehydrate. Give your child an oral rehydration solution, available at pharmacies without a prescription. Talk to your doctor if you have questions about how to use it. Dont give your child plain water — in children with gastroenteritis, water isnt absorbed well and wont adequately replace lost electrolytes. Avoid giving your child apple juice for rehydration — it can

Fig 10: Home remedies given by model

Chapter 4: Conclusion

The integration of machine learning techniques into healthcare systems has ushered in a transformative era, offering unprecedented opportunities for disease prediction and prevention. The conventional healthcare paradigm, reliant on manual analysis and historical data, faces limitations in accuracy and efficiency, which machine learning aims to overcome. The increasing availability of healthcare data and advancements in computational capabilities have fuelled a shift toward leveraging machine learning for disease prediction.

Machine learning, a process of programming computers to improve their output based on examples or previous data, provides a robust framework in the medical sector for efficiently resolving healthcare issues. It employs various methods such as statistics, probabilities, and optimizations to identify crucial patterns in large, unstructured datasets. Most applications rely on supervised learning, where the system learns from labelled data to make predictions about new, unlabelled data.

Early detection of diseases is crucial for improving patient outcomes and reducing healthcare costs. Machine learning algorithms have demonstrated remarkable potential in enhancing the accuracy of disease prediction models by processing diverse patient data, including genetic information, medical imaging, electronic health records, and lifestyle factors. This holistic approach enables more comprehensive prediction and diagnosis.

The motivation behind this research lies in comprehensively understanding the current state-of-the-art in disease prediction using machine learning, identifying successful methodologies, and addressing existing challenges. Despite the success of supervised machine learning in predicting diseases, there is a lack of comprehensive research, particularly in reviewing articles using various supervised learning methods for disease prediction. This study aims to fill that gap by examining trends in different types of supervised machine learning algorithms, their performance, and the focus on specific diseases. The findings will guide researchers in understanding current trends and popular areas in disease prediction, helping them set research goals.

References

[1] Chattopadhyay. A.. Mishra. S.. González-Briones. A. (2021). Integration of Machine Learning and IoT in Healthcare Domain. In: Kumar Bhoi. A.. Mallick. P.K.. Narayana Mohanty. M.. Albuquerque. V.H.C.d. (eds) Hybrid Artificial Intelligence and IoT in Healthcare. Intelligent Systems Reference Library. vol 209. Springer. Singapore.

[2] Balakrishna, S., Thirumaran. M.. Solanki. V.K. (2020). IoT Sensor Data Integration in Healthcare using Semantics and Machine Learning Approaches. In: Balas. V.. Solanki. V.. Kumar, R.. Ahad, M. (eds) A Handbook of Internet of Things in Biomedical and Cyber Physical System. Intelligent Systems Reference Library. vol 165. Springer. Cham.

[3] Uddin. S.. Khan. A.. Hossain. M. et al. Comparing different supervised machine learning algorithms for disease prediction. BMC Med Inform Decis Mak 19. 281 (2019).

[4] T. M. Mitchell. "Machine learning WCB": McGraw-Hill Boston. MA:. 1997.

[5] Sinclair C. Pierce L, Matzner S. An application of machine learning to network intrusion detection. In: Computer Security Applications Conference. 1999. (ACSAC'99) Proceedings. 15th Annual: 1999. p. 371–7. IEEE.

[6] Sahami M. Dumais S. Heckerman D. Horvitz E. A Bayesian approach to filtering junk e-mail. In: Learning for Text Categorization: Papers from the 1998 workshop. vol. 62: 1998. p. 98–105. Madison. Wisconsin.

[7] Aleskerov E. Freisleben B. Rao B. Cardwatch: A neural network based database mining system for credit card fraud detection. In: Computational Intelligence for Financial Engineering (CIFEr). 1997.. Proceedings of the IEEE/IAFE 1997: 1997. p. 220–6. IEEE.

[8] Mahadevan S. Theocharous G. "Optimizing Production Manufacturing Using Reinforcement Learning." in FLAIRS Conference: 1998. p. 372–7.

[9] Yao D. Yang J. Zhan X. A novel method for disease prediction: hybrid of random forest and multivariate adaptive regression splines. J Comput. 2013:8(1):170–7.

[10] Uddin. S.. Khan. A.. Hossain. M. et al. Comparing different supervised machine learning algorithms for disease prediction. BMC Med Inform Decis Mak 19. 281 (2019).

[11] Uddin. S.. Khan. A.. Hossain. M. et al. Comparing different supervised machine learning algorithms for disease prediction. BMC Med Inform Decis Mak 19. 281 (2019).

[12] Takura. T.. Hirano Goto. K. & Honda. A. Development of a predictive model for integrated medical and long-term care resource consumption based on health behaviour: application of healthcare big data of patients with circulatory diseases. BMC Med 19. 15 (2021).

[13] McKinney. B.A.. Reif. D.M.. Ritchie. M.D. et al. Machine Learning for Detecting Gene-Gene Interactions. Appl-Bioinformatics 5. 77–88 (2006).

[14] J. Latif. C. Xiao. A. Imran and S. Tu. "Medical Imaging using Machine Learning and Deep Learning Algorithms: A Review." 2019 2nd International Conference on Computing. Mathematics and Engineering Technologies (iCoMET). Sukkur. Pakistan. 2019

[15] Wong. J.. Murray Horwitz. M.. Zhou. L.. et al. Using Machine Learning to Identify Health Outcomes from Electronic Health Record Data. Curr Epidemiol Rep 5. 331–342 (2018).

[16] Soufiane Ajana. Audrey Cougnard-Grégoire. Johanna M. Colijn. Bénédicte M.J. Merle. Timo Verzijden. Paulus T.V.M. de Jong. Albert Hofman. Johannes R. Vingerling. Boris P. Hejblum, Jean-François Korobelnik. Magda A. Meester-Smoor. Marius Ueffing. Hélène Jacqmin-Gadda. Caroline C.W. Klaver. Cécile Delcourt. Erkin I. Acar. Blanca Arango-Gonzalez. Angela Armento. Franz Badura. Vaibhav Bhatia. Shomi S. Bhattacharya. Marc Biarnés. Anna Borrell. Sofia M. Calado. Sascha Dammeier. Anita de Breuk. Berta De la Cerda. Anneke I. den Hollander. Francisco J. Diaz-Corrales. Sigrid Diether. Eszter Emri. Tanja Endermann, Lucia L. Ferraro. Míriam Garcia. Thomas J. Heesterbeek, Sabina Honisch, Carel B. Hoyng, Ellen Kilger. Elod Kortvely. Claire Lastrucci, Hanno Langen, Imre Lengyel. Phil Luthert, Jordi Monés. Everson Nogoceke. Tunde Peto. Frances M. Pool. Eduardo Rodriguez-Bocanegra. Luis Serrano. Jose Sousa. Eric Thee. Marius Ueffing. Karl U. Ulrich Bartz-Schmidt. Markus Zumbansen. Predicting Progression to Advanced Age-Related Macular Degeneration from Clinical. Genetic. and Lifestyle Factors Using Machine Learning. Ophthalmology. Volume 128. Issue 4. 2021. Pages 587-597

[17] An. Q.: Rahman. S.: Zhou. J.: Kang. J.J. A Comprehensive Review on Machine Learning in Healthcare Industry: Classification. Restrictions. Opportunities and Challenges. Sensors 2023. 23. 4178.

[18] Riccardo Miotto. Fei Wang. Shuang Wang. Xiaoqian Jiang. Joel T Dudley. Deep learning for healthcare: review. opportunities and challenges. Briefings in Bioinformatics. Volume 19. Issue 6. November 2018. Pages 1236–1246

[19] M. M. Kamruzzaman. "New Opportunities. Challenges. and Applications of Edge-AI for Connected Healthcare in Smart Cities." 2021 IEEE Globecom Workshops (GC Wkshps). Madrid. Spain. 2021

[20] Palaniappan S. Awang R. Intelligent heart disease prediction system using data mining techniques. In: Computer Systems and Applications. 2008. AICCSA 2008. IEEE/ACS International Conference on: 2008. p. 108–15. IEEE.

[21] Breiman L. Random forests. Mach Learn. 2001:45(1):5–32.

[22] Joachims T. Making large-scale SVM learning practical. SFB 475: Komplexitätsreduktion Multivariaten Datenstrukturen. Univ. Dortmund, Dortmund, Tech. Rep. 1998. p. 28.