iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** V **Month of publication:** May 2023

DOI: https://doi.org/10.22214/ijraset.2023.51909

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue V May 2023- Available at www.ijraset.com

Multiple Disease Detection Web App

Madikonda Jagadish[1], Anudeep Pulluri[2], Aryan Goud Gollapally[3], M. Yellamma[4]
[1, 2, 3]*B.Tech Scholars, Department of Computer Science and Engineering, SNIST, Hyderabad-501301, India*
[4]*Assistant Professor, Department of Computer Science and Engineering, SNIST, Hyderabad-501301, India*

Abstract: ***Artificial intelligence and machine learning are quite significant in the modern world. They are present in every sector, including the medical sector and autonomous vehicles. The medical industry produces vast amounts of patient data, which is then processed in a variety of ways. Consequently, we were able to create a prediction system that can recognize numerous diseases at once using machine learning. Many of the methods in use now can predict only one disease at a time, and that too with reduced accuracy. Reduced precision can seriously jeopardize a patient's health. The three illnesses we are now considering are diabetes, liver, and heart, but there may be many more added in the future. Several parameters must be entered by the user. When the user enters many disease-related parameters, the system will output whether the user has the disease or not. This initiative can help a lot of people because it makes it possible to monitor someone's health and take the appropriate steps to extend their lives.***

I. INTRODUCTION

In the digital age, data is a valuable resource, and enormous amounts of data were generated across all businesses. The data for the healthcare industry includes all patient-related information. A general framework for disease prediction in the healthcare industry has been presented here. Most of the current models analyse only one disease at a time. For instance, a single analysis might be done for diabetes, cancer, and skin conditions. There isn't a system in place that can examine numerous diseases simultaneously. We are concentrating on giving customers quick and accurate disease predictions based on the symptoms they enter as a result. Therefore, we provide a technique that can be used to predict various diseases using Django. We'll look at heart, diabetes, and malaria problems in this system. Later, a great deal more diseases might be included. We will build numerous illness prediction systems using Django and machine learning algorithms. Python pickling is used to protect the behavior of the model. This system analysis is significant because it considers every element that contributes to the onset of the diseases under study, enabling more accurate and effective disease detection. The finished model's behavior will be saved in a Python pickle file.

II. EXISTING SYSTEM

Currently, a lot of machine learning models used for healthcare analysis focus on just one disease at a time. For instance, the liver is examined first, followed by tests for lung and cancer problems, respectively. A person must visit various websites if they want to predict multiple illnesses. A single analysis cannot forecast several diseases using the same system. The lesser accuracy of some of the models may have major adverse effects on the patient's health.

When a business wants to examine patient health data, it must use multiple models, which in turn drives up costs and lengthens turnaround times. Some systems in use today take into account very little, which can lead to erroneous findings.

III. PROPOSED SYSTEM

Using multiple disease prediction, it is possible to predict several diseases at once. In order to anticipate diseases, the consumer does not need to visit numerous websites. Diabetes, liver, and heart conditions are the main topics of our attention. Given the connection between the three illnesses to implement different sickness analyses, we will use machine learning techniques and Django.

When making a request for access to this API, the user must provide both the name of the illness and its specifications. Django will make a call to the pertinent model and provide the patient's status.

IV. LITERATURE REVIEW

The focus of the study is diabetes because it is one of the deadliest diseases in the world and because it can cause a wide range of illnesses, including blindness. Since it is easy to predict whether a patient will be ill or not, machine learning techniques were used in this study to identify the diabetic condition. To help patients manage their illness, the researchers hoped to create a system that could reliably identify a patient's diabetes. They compared the precision of four main algorithms used in this study: Decision Tree, Nave Bayes, and SVM.

These algorithms' respective levels of accuracy were 85%, 77%, and 77.3%. They also used the ANN algorithm after the training phase to check whether the disease had been appropriately recognized or not. They compared each model's precision, F1 score support, and accuracy in this instance. The accuracy of machine learning for predicting heart disease using k-nearest neighbour, decision tree, linear regression, and SVM is therefore calculated in this work utilizing the UCI repository dataset for training and testing. Additionally, they compared the SVM (83%), Decision Tree (79%), Linear Regression (78%), and K-Nearest Neighbour (87%) algorithms' accuracy. The system states that liver disease is a leading cause of death in India and is also regarded as a serious health issue on a global scale. because it might be challenging to detect liver disease in its early stages. Therefore, using automated software and machine learning algorithms, we can precisely detect liver disease. They employed and compared the SVM, Decision Tree, and Random Forest algorithms to assess precision, accuracy, and recall measures quantitatively and analysis helps us to predict and to take preventive steps.

V. SYSTEM ANALYSIS

A. Functional Requirements

1) The patient is able to forecast the disease thanks to the system.

2) The user enters the information for the specific ailment, and the output is displayed based on the trained model of the user input.

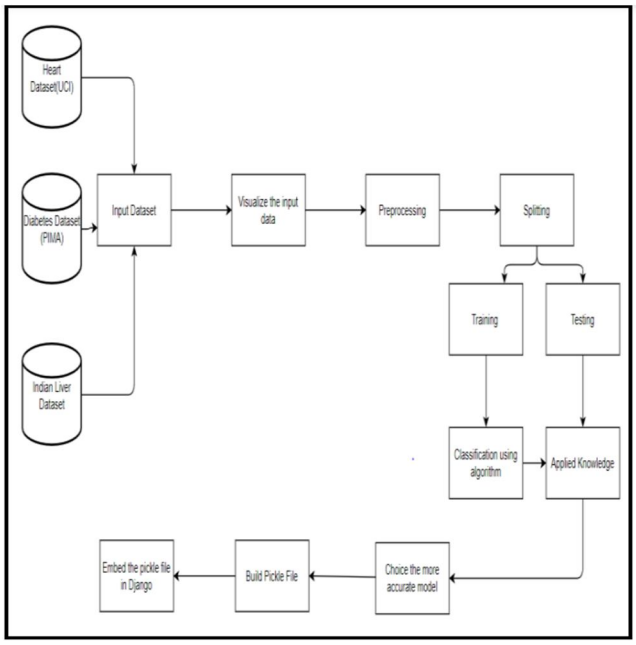
B. Non Functional Requirements

1) The website will provide a range of values when the disease is predicted.

2) The website must be reliable and respectable.

C. Architecture Design

We conducted experiments on three diseases—heart, diabetes, and liver in the below diagram because they are related to one another. The first step is to import the UCI dataset, PIMA dataset, and Indian liver dataset, respectively, for the datasets for heart disease, diabetes disease, and liver disease. After the dataset has been imported, each inputted piece of data is visualized. After pre-processing the data for visualization, which involves looking for outliers, missing values, and scaling the dataset, the data is divided into training and testing. Next, we used the knn, xgboost, and random forest algorithms on the training dataset before applying what we learned about the classified algorithm to the testing dataset. In order to preserve the model's behaviour, Python pickling is employed. The significance of this system analysis is that it considers all the factors that contribute to the development of the diseases under study, making it possible to detect them more effectively and precisely. A Python pickle file will be used to store the behavior of the final model.

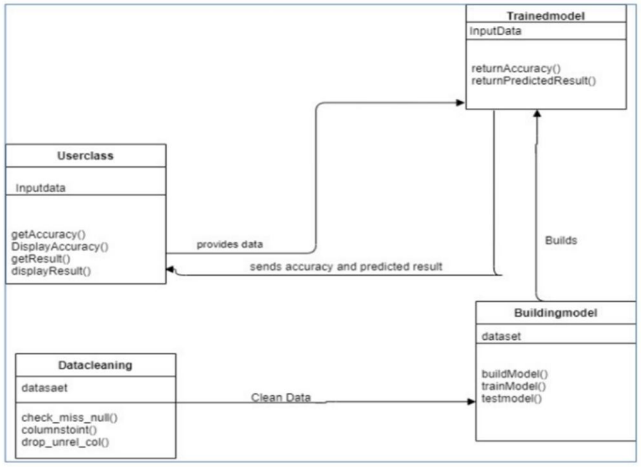


Block Diagram

VI. UML DIAGRAMS

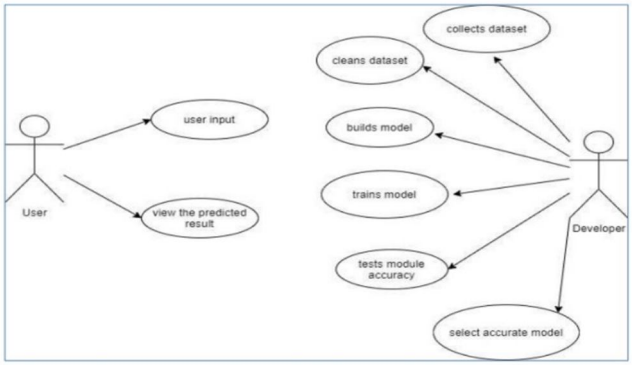
A. Class Diagram

A class is a collection of objects with similar definitions, relationships, functions, and properties. The most important parts of any object-oriented system are classes that implement one or more interface classes.



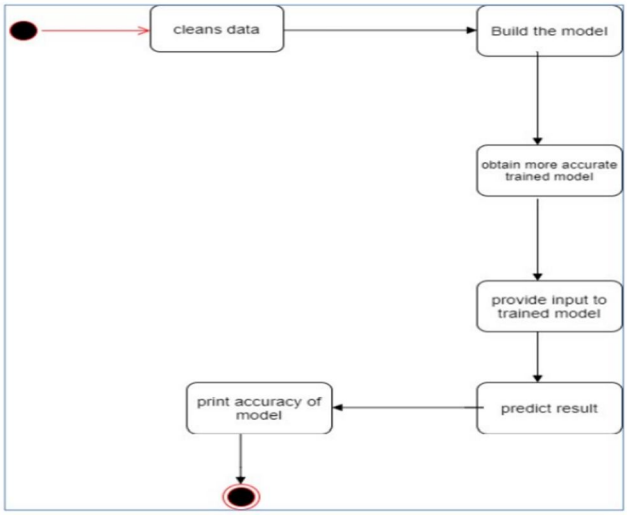
B. Use Case Diagram

The use-case diagram that follows describes the scope and key features of a system. The figure below also shows how the system and its actors interact with one another. Diagrams of use cases demonstrate the function of the system and how actors make use of it.



C. Activity Diagram

The program is represented in the below activity diagram.



D. Technologies Used

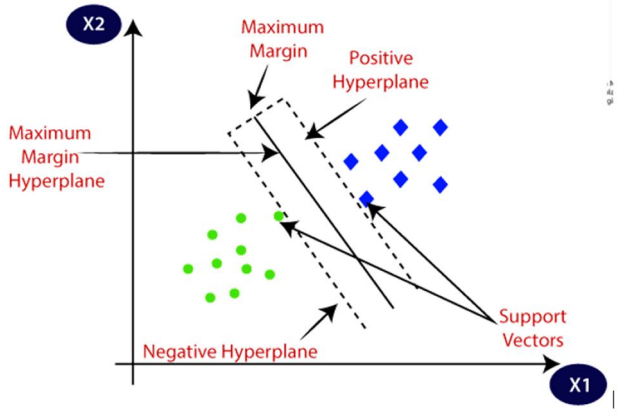
1) Python
- NumPy
- Pandas
- Pickle

2) Streamlit

3) Spyder

E. Algorithm Used

Multiple disorders in the human body can be detected using the SVM algorithm. In order to quickly categorise new data points in the future, the SVM algorithm aims to define the best line or decision boundary that can divide n-dimensional space into classes. The name of this best decision boundary is a hyperplane. The extreme vectors and points that help create the hyperplane are chosen via SVM. The Support Vector Machine approach gets its name from the support vectors that are utilised to represent these extreme occurrences.



Support Vector Machine Algorithm

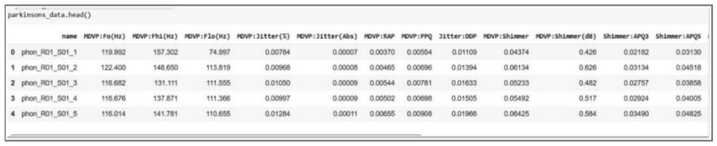
VII. IMPLEMENTATION

The data is taken in the form of a CSV file. We fit the SVM classifier to the training data set and we import the libraries like numpy, pandas, and pickle and then split the data as training and testing data, and then apply the SVM algorithm on the data as a result we get the accuracy score on training data as well as testing data.

A. Code

1) Printing the first five records

```
loaddata.head()
```



2) Count of patients

```
loaddata['status'].value_counts()
```

Output:

```
1    147
0     48
Name: status, dtype: int64
```

3) Grouping data through target variable

```
8] loaddata.groupby('status').mean()
```

status	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	MDVP:Shimmer(dB)
0	181.937771	223.636750	145.207292	0.003866	0.000023	0.001925	0.002056	0.005776	0.017615	0.162958
1	145.180762	188.441463	106.893558	0.006989	0.000051	0.003757	0.003900	0.011273	0.033658	0.321204

2 rows × 22 columns

4) Grouping unwanted columns

```
X = loaddata.drop(columns=['name','status'], axis=1)
Y = loaddata['status']
```

```
[10] #print(X)
```

```
[11] print(Y)
```

Output:

```
[11] print(Y)

0      1
1      1
2      1
3      1
4      1
      ..
190    0
191    0
192    0
193    0
194    0
Name: status, Length: 195, dtype: int64
```

5) Splitting the data into training and testing data

```
[12] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

```
[13] print(X.shape, X_train.shape, X_test.shape)

(195, 22) (156, 22) (39, 22)
```

6) Standardizing the data through SVM

```
[18] model = svm.SVC(kernel='linear')
```

```
[19] # training the SVM model with training data
     model.fit(X_train, Y_train)

     SVC(kernel='linear')
```

```
[20] # accuracy score on training data
     X_train_prediction = model.predict(X_train)
     training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
```

```
[21] print('Accuracy score of training data : ', training_data_accuracy)
```

7) Checking the training and testing data accuracy

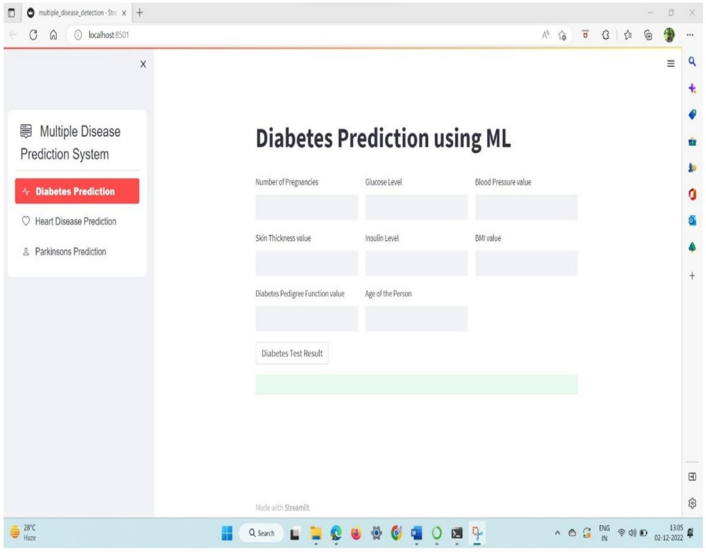
```
[21] print('Accuracy score of training data : ', training_data_accuracy)

     Accuracy score of training data :  0.8846153846153846

[22] # accuracy score on training data
     X_test_prediction = model.predict(X_test)
     test_data_accuracy = accuracy_score(Y_test, X_test_prediction)

[23] print('Accuracy score of test data : ', test_data_accuracy)

     Accuracy score of test data :  0.8717948717948718
```

8) Predictive method

```
[24] input_data = (119.99200,157.30200,74.99700,0.00784,0.00007,0.00370,0.00554,0.01109,0.04374,0.42600,0

     # changing input data to a numpy array
     input_data_as_numpy_array = np.asarray(input_data)

     # reshape the numpy array
     input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

     # standardize the data
     std_data = scaler.transform(input_data_reshaped)

     prediction = model.predict(std_data)
     print(prediction)


     if (prediction[0] == 0):
       print("The Person does not have Parkinsons Disease")

     else:
       print("The Person has Parkinsons")
```

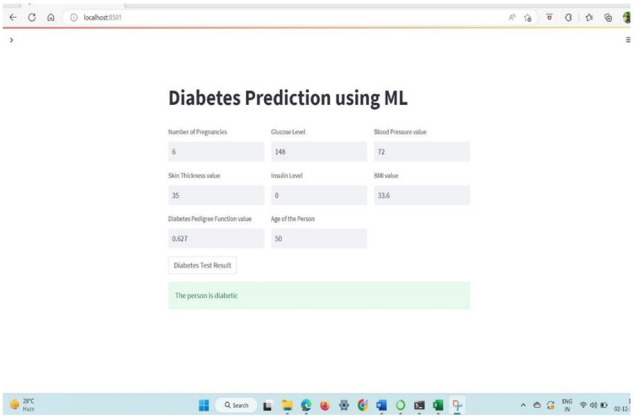
VIII. RESULT/OUTPUT SCREENSHOTS

The below figure is the home page of our project, where we can go through each section of the diseases and the user needs to enter the various fields and mentioned.



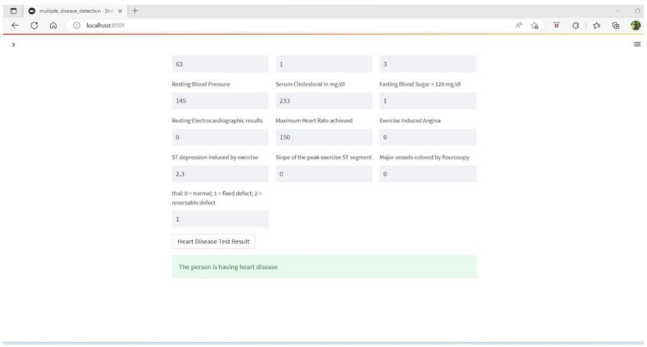
A. Diabetes Page

This user interface page is to predict whether the person has diabetes or not, we need to enter the values for the given fields such as insulin level, age, glucose level, BMI and skin thickness value. . Once after entering the values user need to click on the button for instant result.



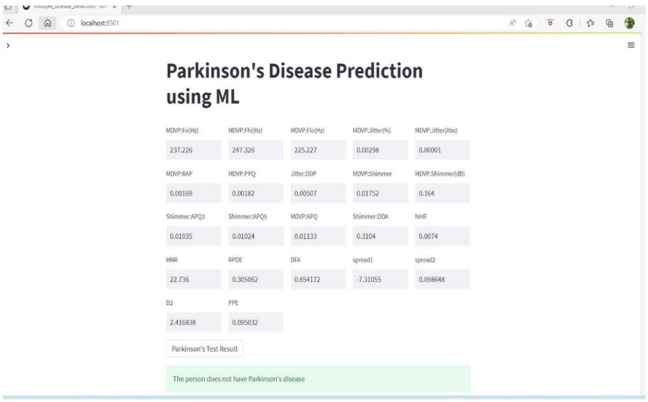
B. Heart Disease

This user interface page is to predict whether the person has heart disease or not, we need to enter the values for the given fields such as resting blood pressure, maximum heart rate achieved, ST depression induced by exercise, the slope of the peak exercise St segment and major vessels covered by fluoroscopy. Once after entering the values user need to click on the button for instant result.



C. Parkinson disease

This user interface page is to predict whether the person has Parkinson's disease or not, we need to enter the values for the given fields such as DFA, spread1, spread2, RPDE, HNR, D2, PPE, Shimmer:DDA, Shimmer:APQ5, Shimmer: APQ3, NHR, MDVP: RAP, MDVP:PPQ, Jitter: DDP values. Once after entering the values user need to click on the button for instant result.



IX. CONCLUSION

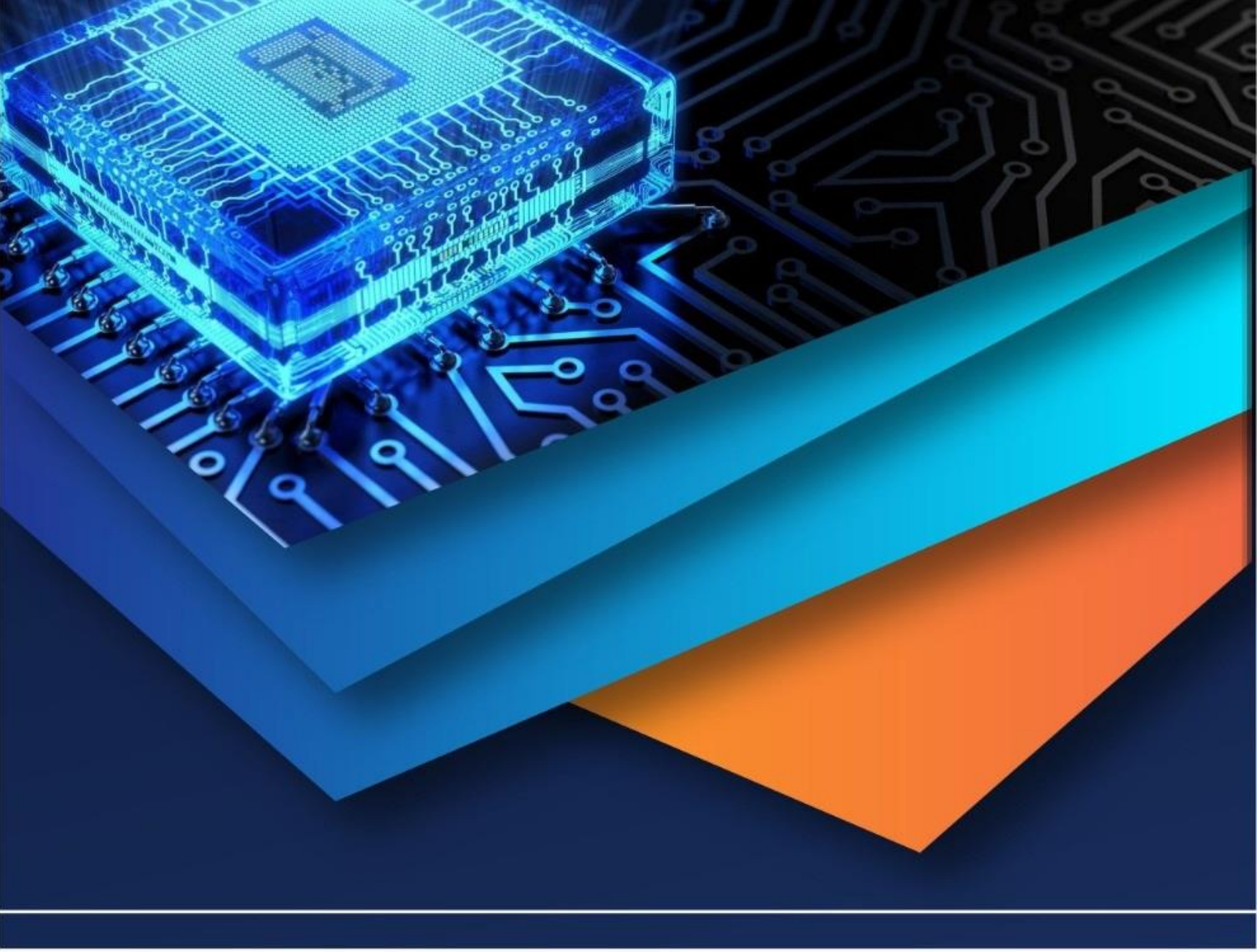
In order to utilize the online application, the user must enter information that will be stored in a model of a flask and used as input by taught machine learning algorithms. The program will detect the multiple diseases of the patient by giving the input values correct. Here, we employed the more accurate Random Forest and svm classification methods. Finally, the output was displayed in a new html page, saying things like "You are healthy" if the patient is fine otherwise the page will be displayed with a serious problem.The program has been thoroughly tested and debugged.

X. FUTURE SCOPE

In order to lower the mortality rate, we can add more diseases to the current API and work to increase forecast accuracy.

REFERENCES

[1] "DIABETES PREDICTION USING DIFFERENT MACHINE LEARNING APPROACHES", 2019 IEEE, 3rd International Conference on Computing Methodologies and Communication (ICCMC), by Priyanka Sonar and Prof. K. JayaMalini.
[2] D. W. Bates, S. Saria, L. Ohno-Machado, A. Shah, and G. Escobar, "Big data in health care: using analytics to identify and manage high-risk and high-cost patients," Health Affairs, vol. 33, no. 7, 2018; pp. 1123–1131.
[3] "Predicting Disease By Using Data Mining Based on Healthcare Information System" by Feixiang Huang, Shengyong Wang, and Chien-Chung Chan was published in IEEE 2012.
[4] "Diagnosis of Liver Disease Using Machine Learning Models" was presented at the 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud; I-SMAC). Authors include A.Sivasangari, Baddigam Jaya Krishna Reddy, Annamareddy Kiran, and P. Ajitha.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY