A NOVEL APPROACH to apply Different Algorithms to Predict COVID-19 Disease

A minor project report submitted in partial fulfilment.

of the requirements for the award of the degree of

Bachelor of Technology

In

COMPUTER SCIENCE AND ENGINEERING

By

170031326	**UTLAPALLI MAHESH**
170030161	**BONELA SYAM JASON**
170031524	**SANKA NITHYA TANVI NISHITHA**

Under the esteemed guidance of

J. SURYA KIRAN ***M.Tech.***

Assistant Professor

Koneru Lakshmaiah Education Foundation

(Deemed to be University estd., u/s 3 of UGC Act 1956)

Greenfields, Vaddeswaram, Guntur (Dist.), Andhra Pradesh - 522502

November, 2019

K L (Deemed to be) University

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

DECLARATION

The Minor Project Report entitled "**A NOVEL APPROACH to apply Different Algorithms to Predict COVID-19 Disease**" is a record of bonafide work of UTLAPALLI MAHESH – 170031326, BONELA SYAM JASON – 170030161 and SANKA NITHYA TANVI NISHITHA - 170031524 submitted in partial fulfilment for the award of **B.Tech.** in "Computer Science and Engineering" during academic year 2020-21 Even Semester.

170031326	**UTLAPALLI MAHESH**
170030161	**BONELA SYAM JASON**
170031524	**S NITHYA TANVI**

Place: Vaddeswaram

Date:

K L (Deemed to be) University

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

CERTIFICATE

This is to certify that the project report entitled "**A NOVEL APPROACH to apply Different Algorithms to Predict COVID-19 Disease**" is a record of bonafide work of UTLAPALLI MAHESH – 170031326, BONELA SYAM JASON – 170030161 and SANKA NITHYA TANVI NISHITHA - 170031524 submitted in partial fulfilment for the award of **B.Tech.** in "Computer Science and Engineering" during academic year 2020-21 Even Semester.

Mr. J. SURYA KIRAN
PROJECT GUIDE
DEPARTMENT OF CSE
KLEF

Mr. HARIKIRAN VEGE
HEAD OF DEPARTMENT
DEPARTMENT OF CSE
KLEF

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

Our sincere thanks to the project guide, **J. Surya Kiran** for his outstanding support throughout the project for the successful completion of the work.

We express our gratitude to **Hari Kiran Vege**, Head of the Department for Computer Science and Engineering, for providing us with adequate facilities, ways, and means by which we can complete this term minor project work.

We would like to place on record the deep sense of gratitude to the Honorable Vice-Chancellor, K L University for providing the necessary facilities to carry the concluded minor project work.

Last but not the least we thank all Teaching and Non-Teaching Staff of our department and especially my classmates and my friends for their support in the completion of our work.

170031326 UTLAPALLI MAHESH

170030161 BONELA SYAM JASON

170031524 S NITHYA TANVI

Place: Vaddeswaram

Date:

ABSTRACT

The Covid-19 episode happened in Wuhan in December 2019 and spread wherever on the planet. The Covid-19 transferable infection has an immunization and medication for its treatment. The premier significant components in lessening the spread of the infection are washing hands, utilizing a veil, and diminishing social distance. Today furthermore to clinical investigations, PC-supported examinations likewise are generally regulated for the Covid-19 episode. Computer-based intelligence strategies are effectively applied during this examination. In this investigation, we used different algorithms for the prediction and analysis of Covid-19 everyday cases. Because of the examination, the number of everyday cases was effectively assessed with these various kinds of algorithms.

CONTENTS

1. INTRODUCTION

The novel Coronavirus (COVID-19) has infected millions of people worldwide since it emerged from China in December 2019. COVID-19 has very high mutating capability, and it can spread very easily. Infected people from this virus suffer from severe respiratory problems and may develop serious illness if suffering from chronic diseases like cardiovascular disease or diabetes or having weak immune system or being older in age. World health organization (WHO) declared on 11th March 2020, the outbreak of COVID-19 as a pandemic. There are challenges to contain the disease because an infected person shows symptom after a long time or no sign of the disease. At present, no vaccination has been discovered for COVID-19. In this situation, social distancing, identifying the positive cases using testing at large scale, and containment of infected person is the only option to prevent the spreading of the virus. The spread of COVID-19 [20] can be classified under three major stages- 1. Local outbreak: at this stage, spreading chain of the virus among the people can be tracked, and the source of infection can be found out. The cases in this stage mostly relate to within family or friends, or the local exposure. 2. Community transmission: at this stage, source of the chain of infected people cannot be found out. The infected cases grow through cluster transmission in the communities. 3. Large scale transmission: at this stage, the virus spreads rapidly to other regions of a country due to uncontrolled mobility of people at large scale. Due to high scale community impact and easy spreading worldwide, national governments-imposed lockdown to control the spread of corona virus. As of 20th May 2020, 4996472 cases have been confirmed, 1897466 cases have recovered, 2328115 deaths have been reported, and 2770891 active cases have been identified worldwide. The statistical data is collected from, and the number of COVID-19 [11] cases is calculated between 22 Jan 2020 to 20 May 2020. As no vaccine has been discovered of the disease, so motivation behind this paper is to model spreading of the corona virus and predict the impact to optimize the planning to manage the various services and resources for the public by the governments. Some of the studies have been published showing statistical analysis, modeling, and artificial intelligence to contain the spread of the virus and highlight impacts in coming days. These early studies are carried out using very limited information available at early stage of the outbreak. Now, the virus has spread at large scale, and much information is available for the analysis. Predictive analysis of COVID-19 has become a hot research area to support health services and governments to plan and contain the spread of the infectious disease. Modeling and forecasting the daily spread behavior of the virus can assist the health

systems to be ready to accommodate the upcoming number of patients. Accurate forecasting of the disease is a matter of concern because it may impact governments policy, containment rules, health system, and social life. Regarding this context, we explore the predictive capability of the ARIMA, and Prophet forecasting models. The models are widely used and accepted due to their more accurate forecasting capability. We use the day level cumulative cases of COVID-19 worldwide and 10 mostly affected countries, US, Spain, Italy, France, Germany, Russia, Iran, United Kingdom, Turkey, and India for our analysis study.

2. PROBLEM STATEMENT

Coronavirus disease (COVID-19) is an inflammation disease from the latest virus. This particular infection causes respiratory ailment (like influenza) with manifestations, for example, cold, cough, and fever, and in progressively serious cases, The problem in breathing. COVID-2019 has been perceived as a worldwide pandemic and a couple of examinations are being led utilizing different numerical models to anticipate the likely advancement of this pestilence. These numerical models hooked into various factors and investigations are dependent upon potential inclination. Here, we presented a model that could be useful to predict the Covid-19 daily cases by using different algorithms.

3. LITERATURE SURVEY

i) Title: Covid Symptom Severity Using Decision Tree

Authors: Naim Rochmawati, Hanik Badriyah Hidayati, Yuni Yamasari, Wiyli Yustanti, Lusia Rakhmawati, Hapsari P. A. Tjahyaningtijas, Yeni Anistyasari

In this paper, the main aim is to present a simple key whether it is symptoms of covid-19 [21] or normal cough. A clinical symptom dataset has been taken which is used for classifying the symptoms using the Decision Tree algorithm. In this research, the decision tree that has been used is hoeffding and j48. As we know that decision is one of the widely used algorithms for classification methods [12] as it is easy to simply by humans. the main aim is to cover the concept of concerting data into decision trees or decision rules. As a result, j48 has shown more efficiency than hoeffding tree in terms of recall, accuracy, precision. In further, this research work can be also used for the same dataset and different preprocessing. [1]

ii) Title: COVIDGR Dataset and COVID-SDNet Methodology for Predicting COVID-19 Based on Chest X-Ray Images

Authors: S. Tabik, A. Gómez-Ríos, J. L. Martín-Rodríguez, I. Sevillano-García, M. Rey-Area, D. Charte, E. Guirado, J. L. Suárez, J. Luengo, M. A. Valero-González, P. García-Villanova, Olmedo-Sánchez, and F. Herrera

Current pandemic covid-19 is detected by taking the help of CT-Scans, X-Rays, CT scans. As we know that RT-PCR tests are mostly not available in medical centres and pharmacy stores hence CXR images have been used alternatively which is the most efficient, less cost, less time tool for assisting in taking the decisions. Here DL Neural networks have been used and shown potential for building the covid-19 triage systems and to identify the covid-19 patients mostly detected patient with less severity. this research around two points i.e., clarify the high sensitivities achieved in covid-19 classification models, well collaborate with most hospitals such as Spain, Granada. The dataset is stabilized by using the covidgr-1.0 that has every level. To use it in the future, it can enhance with covidgr-1.0 with a CXR image taken from multiple hospitals. [2]

iii) Title: A Novel Parametric Model for the Prediction and Analysis of the COVID-19 Casualties

Authors: ONDER TUTSOY, ŞULE ÇOLAK, ADEM POLAT, AND KEMAL BALIKCI

This papers mainly cover the concepts of multi-dimensional, parametric suspicious infected death, strongly coupled model. The research on mathematical analytics has shown the results on using region, instability region, stable region. the information has been analysed and lifting the restriction shows that the covid-19 is moderately unstable which in the future will increase if no actions are taken. the model evaluates that many dead and contaminated will meet zero in 300 days, but the number of uncertainty individuals requires 1000 days to decrease under the present conditions. Allowing the model was used to evaluate the corresponding covid-19 casualties. So, in further, it can be used more efficiently without taking the uncertainty, death causalities, infected into consideration. we must also avoid pharmacological policies, non-pharmacological policies, and intubation casualties. [3]

iv) Title: Prediction of Covid-19 pandemic based on Regression

Authors: Ashish U Mandayam, Rakshith.A.C, Siddesha S, S K Niranjan

As we are in the COVID-19 pandemic, it would be useful to anticipate the forthcoming number of positive cases for better assessment and control. We have chipped away at two managed learning models to gauge the future utilizing the time-arrangement dataset of COVID-19. To examine the arranging of the forecast, the correlation between Linear Regression [13] and Support Vector Regression is performed. We chose to utilize these two models as the information was almost straight. at the point when we make progress toward the improvement of our outcomes, we came to see that SVR contrasted with Linear Regression with time-arrangement information, the Linear Regression calculation performs better since the informational collection utilized here is likewise straight and the SVR can't deal with enormous direct datasets quite well. [4]

v) Title: Covid-19 Disease Simulation using GAMA platform

Authors: *Tran Quy Ban, Phan Lac Duong, Nguyen Hoang Son, Tran Van Dinh*

As the Covid-19 pandemic is advancing, information is gathered from different sources. From one perspective, specialists permit to make educated acclimations to the current and arranged intercessions and uncover them. Then again, a dire requirement for devices and strategies that empower speedy investigation, getting, correlation, and anticipating of the viability of the reactions against COVID-19 across various networks and settings. In this point of view, computational displaying shows up as priceless influence as it permits us to investigate in silico an assortment of mediation procedures before the likely period of field execution. For future work, we intend to advance the reproduction with more functionalities. We additionally need to think about among GAMA and other recreation structures to have an appropriate view of the re-enactment system as a rule. [5]

vi) Title: A Novel Deep Learning Approach for Classification of COVID-19 Images

Authors: Malaya Kumar Nath, Aniruddha Kanhe, Madhusudhan Mishra

In this paper, the capability of man-made brainpower [19] has been researched to build up a profound neural organization model for fast, exact, and compelling COVID-19 location from the CT and X-beam pictures. The proposed strategy gives a strong profound learning method to parallel (COVID versus NON-COVID) and multi-class (COVID versus NON-COVID versus Pneumonia) grouping from X-ray and CT pictures. A 24-layer CNN network has been proposed for the grouping. It achieves an exactness of 99.68% and 71.81% on X-beam and CT pictures, individually. For both datasets, the Sgdm streamlining agent has been utilized with a learning rate of 0.001. As in the future to enhance this we will make These simulations to work on almost every system. [6]

vii) Title: Regression Analysis of COVID-19 using Machine Learning Algorithms

Authors: Ekta Gambhir, Ritika Jain, Alankrit Gupta, Uma Tomer

This paper means to supply much better comprehension of how different Machine Learning [14] models can be executed in true circumstances. Aside from the examination done in the world figures, this paper additionally investigations the current pattern or example of Covid-19 transmission in India. With the assistance of datasets from the Ministry of Health and Family Welfare of India, this work advances different patterns and examples experienced in various pieces of the world. The information to be read has been gotten for 154 days i.e., from January 22, 2020, till June 24, 2020. For future reference, the information can be additionally investigated, and more outcomes can be acquired. [7]

viii) Title: Fuzzy Rule-Based System for Predicting Daily Case in COVID-19 Outbreak

Authors: Pinar Cihan

As we know that covid-19 is a new virus and hasn't proven clinically or had a drug/vaccine for the treatment. A most prime factor to reduce the increase of virus is to maintain Social distance, always wearing up masks (N95 preferably), sanitizing hands regularly. As we know today's computer-aided technology is commonly used for covid-19. In that AI [18] is one of the best successful applications in epidemic studies. In this paper, fuzzy concepts have been used such as FRBs which is used for prognostication. The number of covid cases increasing daily. As result the number of covid cases has been successfully estimated i.e., R2 = 0.96, MAE = 186, RMSE = 254. [8]

ix) Title: COVID-19 Pandemic Prediction using Time Series Forecasting Models

Authors: Naresh Kumar,Seba Susan

In this paper, the aim is to identify the overlong infected covid cases and the growth of virus spread is evaluated to reduce and take the precautions within the health care service ignorer to avoid the deaths. As we know that predicting the growth/speed of covid-19 is the most challenging real-world problem hence here the day level information has been taken from the top 10 most affected covid-19 countries such as Spain, us, Italy, Germany, Iran, Russia, UK,

Turkey, India. The data has been taken from January 22, 2020, to May 20, 2020. Then according to the data, the model of evaluation of covid-19 is predicted by models such ARIMA and prophet time series forecasting [16]. The absolute mean error, root mean square error, root relative squared error, and mean absolute percentage error also has been calculated. From the output understand that the ARIMA model is suitable for prognosticating [17] covid-19 similarity. [9]

x) Title: A Comparative Study of Machine Learning Models for COVID-19 prediction in India

Authors: Vartika Bhadana, Anand Singh Jalal, Pooja Pathak

The paper shows the capacity of ML models to evaluate the proportion of approaching COVID-19-affected patients that is as of now saw as a certified risk to human progress. During this paper, we have administered comparative research on five machine learning [15] standard models like LR, decision tree, LASSO, random forest, and SVM to forecast the frightening variables of COVID-19. Each of the models makes three sorts of forecasts, i.e., the total active cases, the entire deaths, and the total recoveries within the next five days. The findings provided by the paper recommend that the utilization of those techniques for the present COVID-19 pandemic scenario may be a promising strategy. For greater accuracy, we have used a six-degree polynomial. Experiment outputs illustrate that poly LR, and poly LASSO gives the best results followed by LR, LASSO, random forest, and decision tree. SVM shows a poor outcome in the prediction of COVID-19. [10]

4. EXISTING SYSTEM

The Further, more attributes can be included in the study to add more accuracy during the process. Further investigation may involve the analysis on India dataset by predicting the number of cases in future and how the mortality rate varies with the rise in the number of cases. Hope this article contributes to the world's response to this epidemic and puts forward some references for further research in future. Many Existing Systems are used only one or two algorithms, but they have not compared with other algorithms.

4.1 Disadvantages of Existing System

- Dataset is having only small number of attributes for prediction
- These simulations do not work on all systems.
- Forecasting is being done on some factors only.
- These datasets having a smaller number of images, which may affect the proposed network accuracy for a large dataset.
- Different datasets are using in the regression analysis.
- It is difficult to model physical systems precisely with a mathematical formula or equation.

5. PROPOSED SYSTEM

In the proposed system we are analysing and predicting the Covid-19 daily cases based upon the different factors. Here we are using different types of algorithms for the analysis of the Covid-19 increasing cases and predicting the spreading rate of Covid-19 cases based on the previous diseases history of the person with the help of training data and the testing data. By using those different types of algorithms, we can achieve the highest accuracy rate when compared with other algorithms.

5.1 Advantages of Proposed System

- We can take Dataset having many numbers of attributes for prediction
- These simulations work on all systems.
- Forecasting is being done on required factors.
- These datasets having a larger number of images, which may not affect the proposed network accuracy for a large dataset.
- It is easy to model physical systems precisely with a mathematical formula or equation.

6. METHODOLOGY

In our Proposed System we have used the below 5 Algorithms to predict COVID-19 disease

i. Decision Tree
ii. K Nearest Neighbors
iii. SVM
iv. Naïve Bayes
v. Random Forest

6.1 Decision Tree:

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.
- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Working of Decision Tree Algorithm:

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using Attribute Selection Measure (ASM).
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the Random Forest algorithm.
- For more class labels, the computational complexity of the decision tree may increase.

6.2 K Nearest Neighbors:

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

Working of KNN Algorithm:

The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of K number of neighbors.
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

6.3 SVM:

- Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

- The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

- SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

Types of SVM

SVM can be of two types:

1. **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
2. **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

6.4 Naïve Bayes:

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in text classification that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
- Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

Working of Naïve Bayes' Algorithm:

Working of Naïve Bayes' Classifier can be understood with the help of the below example:

Suppose we have a dataset of **weather conditions** and corresponding target variable "**Play**". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

Advantages of Naïve Bayes Classifier:

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.
- It performs well in Multi-class predictions as compared to the other Algorithms.
- It is the most popular choice for text classification problems.

Disadvantages of Naïve Bayes Classifier:

- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

6.5 Random Forest:

- Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

- As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.
- The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Working of Random Forest Algorithm:

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

Advantages of Random Forest

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages of Random Forest

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

7. BLOCK DIAGRAM

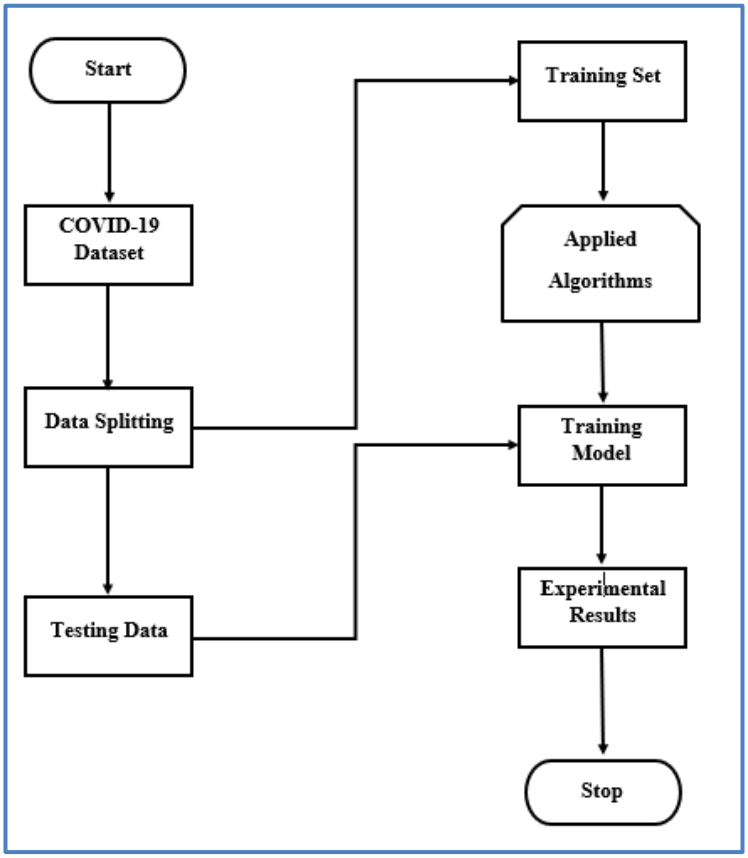

Fig 1: Block Diagram of Proposed System

Algorithm of the Proposed System:

Step 1: Start

Step 2: Required dataset collection using Kaggle and GitHub repositories.

Step 3: Pre-processing the dataset to remove the noisy data.

Step 4: Splitting the data into training and testing data.

Step 5: Creation and training the models using the training data.

Step 6: Calculating accuracy of the models using the test data.

Step 7: Comparison of experimentation results.

Step 8: Stop

8. SOFTWARE REQUIREMENTS

Technologies used are:

Python 3.0 or Above version

Python is platform independence. There are pros as well as cons with the interpreted languages. A high-level programming language class that executes many typical programming behaviors during runtime that static programming languages perform during compilation. Garbage collection alleviates the developer from manual memory control in which the programmer determines which artifacts to view and return to the memory system. It belongs to the class of high-level language. Garbage collection eliminates many bugs like. dangling pointers, double free bugs. It requires computational power when choosing which memory to spare. Python also support functional programming. Functional programming is a program framework where the code and composition functions are used to construct programs. It is a declarative programming model in which function descriptions are trees of expressions that each return a value, instead of a sequence of imperative statements that modify the state of the program of the environment. The basic library of a language is frequently regarded by its users as part of the language although it may have been viewed as a distinct body by the programmers. In addition to other parts that can be optionally implemented, several language standards describe a central collection that must be made accessible in all implementations. Therefore, the line between a language and its libraries varies from language to language. It has the standard library from which all the packages can be installed. There are many upgradations and new features available in the later versions of python. It can be installed in all different operating systems like Windows, macOS, Linux. Python is an open-source software programming language. The python software foundation manages the python software development and all its resources An operating system is machine software that administers computing tools and offers common computer programs services. Time-sharing operating systems plan activities for effective device use, which can include accounting tools for machine time management data storage, printing, and other services Python is supposed to be easy to read the language. Its formatting is physically uncluttered and uses English keywords while punctuation is used by other languages. In many other languages, the curly brackets are not used to delimit lines, and semicolons are available for sentences. It has fewer syntactic variants than C or Pascal and special cases Instead of curly brackets or keywords, Python uses white space indentation to

delimit the blocks. A rise in indentation occurs after certain statements, a reduction in indentation indicates the end of the current row.

8.1 PACKAGES USED

Pandas:

To perform Data analysis and preprocessing of data in python we have a built in "pandas" library, which provides various inbuilt functions that performs operations for manipulating numerical tables and time series.

Matplotlib:

To perform Data Visualization in python, we use Matplotlib which is an inbuilt library that performs 2D visualization of data. Matplotlib generate high quality bar charts, scatter plots, histograms, and many more.

NumPy:

NumPy is a python library that works on Numerical data, which contains a multi-dimensional arrays and matrix data structures. It is mainly used to perform mathematical operations on Numerical data during analysis.

Sklearn:

Scikit-learn in Python is the most useful library that provides efficient tools for machine learning and statistical modeling including classification, regression, and clustering techniques.

Classification_report:

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False? More specifically, True Positives, False Positives, True negatives, and False Negatives are used to predict the metrics of a classification report. The report shows the main classification metrics precision, recall and f1-score on a per-class basis. The metrics are calculated by using true and false positives, true and false negatives. Positive and negative in this case are generic names for the predicted classes.

Accuracy_score:

Accuracy classification score.

In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true.

BaggingClassifier:

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction.

Seaborn:

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

9. EXPERIMENTAL SET UP AND RESULTS

9.1 Dataset

- We have collected the 3617 records from the Kaggle website and some other GitHub repositories.

9.2 Procedure

Step 1: Firstly, the collected dataset is taken and then we undergone some pre-processing techniques to clean the raw data.

Step 2: Then the dataset is split into training and testing data

Step 3: Then we need to apply different types of machine learning algorithms on the training data.

Step 4: Then we need to find the accuracy of the testing data with the help of those models.

Step 5: Lastly, we need to compare the experimental results obtained by those models.

10. IMPLEMENTATION

10.1 Decision Tree Algorithm:

```
import time
import graphviz
import pandas as pd
from sklearn.ensemble import BaggingClassifier
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, export_graphviz
import sklearn.metrics
from sklearn import tree
from sklearn.metrics import accuracy_score
start_time = time.time()
df = pd.read_csv('PREPARED.csv',sep=",")
df
df_numeric_only = df._get_numeric_data()
df_numeric_only
x = df_numeric_only.values[:, 0:4]
y = df_numeric_only.values[:,4]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size= 0.30)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
features = list(df.columns[:4])
#from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = "entropy", random_state = 1, splitter='best')
classifier = classifier.fit(x_train,y_train)
predictions = classifier.predict(x_test)
print(sklearn.metrics.confusion_matrix(y_test, predictions))
```

- print("accuracy of training dataset is{:.2f}".format(classifier.score(x_train,y_train)))
- print("accuracy of test dataset is {:.2f}".format(classifier.score(x_test,y_test)))
- print("Error rate is",1- accuracy_score(y_test, predictions, normalize = True))
- print("sensitivity is", sklearn.metrics.recall_score(y_test, predictions,labels=None, average = 'micro', sample_weight=None))
- print("specificity is", 1 - sklearn.metrics.recall_score(y_test, predictions,labels=None, average = 'micro', sample_weight=None))

10.2 KNN Algorithm:

```
#K Nearest Neighbors with Python
#Import Libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn.metrics
import numpy as np
#Load the Data
df = pd.read_csv("PREPARED.csv",index_col=0)
df.head()
#Standardize the Variables
#Because the KNN classifier predicts the class of a given test observation
#by identifying the observations that are nearest to it, the scale of the
#variables matters. Any variables that are on a large scale will have a much
#larger effect on the distance between the observations, and hence on the KNN
#classifier, than variables that are on a small scale.
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df= df._get_numeric_data()
scaler.fit(df.drop('AFFECTED',axis=1))
scaled_features = scaler.transform(df.drop('AFFECTED',axis=1))
df_feat = pd.DataFrame(scaled_features,columns=df.columns[:-1])
df_feat.head()
#Train-Test Split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
train_test_split(scaled_features,df['AFFECTED'],test_size=0.30)
#Using KNN
#Remember that we are trying to come up with a model to predict whether someone
#will TARGET CLASS or not. We'll start with k=1.
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
```

```
knn.fit(X_train,y_train)
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
metric_params=None, n_jobs=1, n_neighbors=1, p=2,
weights='uniform')
pred = knn.predict(X_test)
#Predicting and evavluations
#Let's evaluate our knn model.
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
#Choosing a K Value
#Let's go ahead and use the elbow method to pick a good K Value:
error_rate = []
# Will take some time
for i in range(1,600):
knn = KNeighborsClassifier(n_neighbors=i)
knn.fit(X_train,y_train)
pred_i = knn.predict(X_test)
error_rate.append(np.mean(pred_i != y_test))
plt.figure(figsize=(10,6))
plt.plot(range(1,600),error_rate,color='blue', linestyle='dashed',
marker='o',markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train,y_train)
pred = knn.predict(X_test)
pred
print('WITH K=1')
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
from sklearn.metrics import accuracy_score
```

- accuracy_score(y_test,pred)
- knn = KNeighborsClassifier(n_neighbors=23)
- knn.fit(X_train,y_train)
- pred = knn.predict(X_test)
- pred
- print('WITH K=23')
- print(confusion_matrix(y_test,pred))
- print(classification_report(y_test,pred))
- print(accuracy_score(y_test,pred))
- print("Error rate is",1- accuracy_score(y_test, pred, normalize = True))
- print("sensitivity is", sklearn.metrics.recall_score(y_test, pred,labels=None, average = 'micro', sample_weight=None))

10.3 SVM Algorithm:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn.metrics
from sklearn.metrics import accuracy_score
df = pd.read_csv('PREPARED.csv',sep=",")
df.tail()
df.shape
df.size
df.count()
df
df_numeric_only = df._get_numeric_data()
df_numeric_only
df.dtypes
x = df_numeric_only.values[:, 0:4]
y = df_numeric_only.values[:,4]
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size= 0.30)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
from sklearn import svm
classifier = svm.SVC(kernel="linear",gamma="auto", C=2)
classifier.fit(x_train,y_train)
y_predict = classifier.predict(x_test)
from sklearn.metrics import classification_report
print(classification_report(y_test,y_predict))
print("accuracy of training dataset is{:.2f}".format(classifier.score(x_train,y_train)))
print("accuracy of test dataset is {:.2f}".format(classifier.score(x_test,y_test)))
print("Error rate is",1- accuracy_score(y_test, y_predict, normalize = True))
```

- print("sensitivity is", sklearn.metrics.recall_score(y_test, y_predict,labels=None, average = 'micro', sample_weight=None))

10.4 Naïve Bayes Algortithm:

```
import time
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
import sklearn.metrics
from sklearn.metrics import accuracy_score
start_time = time.time()
df = pd.read_csv('PREPARED.csv',sep=",")
df
df_numeric_only = df._get_numeric_data()
df_numeric_only
x = df_numeric_only.values[:, 0:4]
y = df_numeric_only.values[:,4]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size= 0.20)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
features = list(df.columns[:4])
gnb = GaussianNB()
y_pred = gnb.fit(x_train, y_train).predict(x_test)
y_pred
print("Number of mislabeled points out of a total %d points : %d"
...       % (x_test.shape[0], (y_test != y_pred).sum()))
gnb.fit(x_train,y_train)
accuracy_score(y_test,y_pred)
accuracy_score(y_train,gnb.predict(x_train))
print("Error rate is",1- accuracy_score(y_test, y_pred, normalize = True))
print("sensitivity is", sklearn.metrics.recall_score(y_test, y_pred,labels=None,
average =  'micro', sample_weight=None))
```

10.5 Random Forest Algorithm:

```
# Random Forest Classifier
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn.metrics
# Importing the datasets
datasets = pd.read_csv('PREPARED.csv')
X = datasets.iloc[:, [2,3]].values
Y = datasets.iloc[:, 7].values
X
Y
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_Train, X_Test, Y_Train, Y_Test = train_test_split(X, Y, test_size = 0.25,
random_state = 0)
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_Train = sc_X.fit_transform(X_Train)
X_Test = sc_X.transform(X_Test)
# Fitting the classifier into the Training set
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 20, criterion = 'entropy',
random_state = 0)
classifier.fit(X_Train,Y_Train)
# Predicting the test set results
Y_Pred = classifier.predict(X_Test)
Y_Pred
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_Test, Y_Pred)
```

- cm
- # Visualising the Training set results
- from matplotlib.colors import ListedColormap
- X_Set, Y_Set = X_Train, Y_Train
- X1, X2 = np.meshgrid(np.arange(start = X_Set[:, 0].min() - 1, stop = X_Set[:, 0].max() + 1, step = 0.01),
    - np.arange(start = X_Set[:, 1].min() - 1, stop = X_Set[:, 1].max() + 1, step = 0.01))
- plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
    - alpha = 0.75, cmap = ListedColormap(('red', 'green')))
- plt.xlim(X1.min(), X1.max())
- plt.ylim(X2.min(), X2.max())
- for i, j in enumerate(np.unique(Y_Set)):
- plt.scatter(X_Set[Y_Set == j, 0], X_Set[Y_Set == j, 1],
    - c = ListedColormap(('red', 'green'))(i), label = j)
- plt.title('Random Forest Classifier (Training set)')
- plt.xlabel('Disease')
- plt.ylabel('Status')
- plt.legend()
- plt.show()
- # Visualising the Test set results
- from matplotlib.colors import ListedColormap
- X_Set, Y_Set = X_Test, Y_Test
- X1, X2 = np.meshgrid(np.arange(start = X_Set[:, 0].min() - 1, stop = X_Set[:, 0].max() + 1, step = 0.01),
    - np.arange(start = X_Set[:, 1].min() - 1, stop = X_Set[:, 1].max() + 1, step = 0.01))
- plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
    - alpha = 0.75, cmap = ListedColormap(('red', 'green')))
- plt.xlim(X1.min(), X1.max())
- plt.ylim(X2.min(), X2.max())

- for i, j in enumerate(np.unique(Y_Set)):
- plt.scatter(X_Set[Y_Set == j, 0], X_Set[Y_Set == j, 1],
    - c = ListedColormap(('red', 'green'))(i), label = j)
- plt.title('Random Forest Classifier (Test set)')
- plt.xlabel('Diseases')
- plt.ylabel('Status')
- plt.legend()
- plt.show()
- from sklearn import metrics
- print('Mean Absolute Error:', metrics.mean_absolute_error(Y_Test, Y_Pred))
- print('Mean Squared Error:', metrics.mean_squared_error(Y_Test, Y_Pred))
- print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(Y_Test, Y_Pred)))
- from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
- print(confusion_matrix(Y_Test, Y_Pred))
- print(classification_report(Y_Test, Y_Pred))
- print(accuracy_score(Y_Test, Y_Pred))
- print("Error rate is",1- accuracy_score(Y_Test, Y_Pred, normalize = True))
- print("sensitivity is", sklearn.metrics.recall_score(Y_Test, Y_Pred,labels=None, average = 'micro', sample_weight=None))

11. OUTPUT SCREENS

```
df
```

	AGE	GENDER	LUNG DISEASE	HEART DISEASE	GASTRIC DISEASE	LOCALITY	STATUS	AFFECTED
0	51	F	1	0	0	CITY	DISEASED	0
1	45	M	1	1	1	CITY	DISEASED	1
2	21	F	0	1	0	CITY	DISEASED	1
3	56	M	0	1	0	VILLAGE	RECOVERED	1
4	41	F	1	0	1	CITY	DISEASED	0
5	15	M	1	0	1	CITY	DISEASED	0
6	12	M	0	1	0	VILLAGE	RECOVERED	1
7	51	F	0	1	1	VILLAGE	DISEASED	1
8	45	M	1	0	0	CITY	DISEASED	0
9	12	F	0	1	1	CITY	DISEASED	1
10	45	M	1	1	0	CITY	RECOVERED	1

Fig 2 Description: The above figure shows the output of the given code.

```
df_numeric_only
```

	AGE	LUNG DISEASE	HEART DISEASE	GASTRIC DISEASE	AFFECTED
0	51	1	0	0	0
1	45	1	1	1	1
2	21	0	1	0	1
3	56	0	1	0	1
4	41	1	0	1	0
5	15	1	0	1	0
6	12	0	1	0	1
7	51	0	1	1	1
8	45	1	0	0	0
9	12	0	1	1	1
10	45	1	1	0	1
11	84	1	0	1	0
12	45	0	1	0	1
13	26	0	1	1	1
14	54	1	0	1	0

3601	40	0	1	0	1
3602	40	0	0	0	0
3603	24	1	1	1	1
3604	70	0	1	1	1
3605	40	0	0	1	0
3606	43	1	0	0	0
3607	45	0	1	0	1
3608	49	0	0	1	0
3609	21	1	1	0	1
3610	47	1	0	1	0
3611	22	1	0	1	1
3612	68	0	1	1	1
3613	31	0	0	0	0
3614	53	1	0	0	0
3615	25	1	1	0	1

3616 rows × 5 columns

Fig 3 Description: The above figure shows the output of the given code.

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(2531, 4)
(1085, 4)
(2531,)
(1085,)
```

Fig 4 Description: The above figure shows the output of the given code.

```
print("accuracy of training dataset is{:.2f}".format(classifier.score(x_train,y_train)))
print("accuracy of test dataset is {:.2f}".format(classifier.score(x_test,y_test)))
```

```
accuracy of training dataset is0.96
accuracy of test dataset is 0.88
```

Fig 5 Description: The above figure depicts the accuracy of the Decision Tree Algorithm.

```
In [113]: print("Error rate is",1- accuracy_score(tar_test, predictions, normalize = True))

          Error rate is 0.09392265193370164
```

Fig 6 Description: The above figure depicts the Error Rate of the Decision Tree Algorithm.

```
#Load the Data

df = pd.read_csv("PREPARED.csv",index_col=0)
df.head()
```

AGE	GENDER	LUNG DISEASE	HEART DISEASE	GASTRIC DISEASE	LOCALITY	STATUS	AFFECTED
51	F	1	0	0	CITY	DISEASED	0
45	M	1	1	1	CITY	DISEASED	1
21	F	0	1	0	CITY	DISEASED	1
56	M	0	1	0	VILLAGE	RECOVERED	1
41	F	1	0	1	CITY	DISEASED	0

Fig 7 Description: The above figure shows the output of the given code.

```
df_feat = pd.DataFrame(scaled_features,columns=df.columns[:-1])
df_feat.head()
```

	LUNG DISEASE	HEART DISEASE	GASTRIC DISEASE
0	1.036048	-0.939858	-0.956689
1	1.036048	1.063990	1.045272
2	-0.965207	1.063990	-0.956689
3	-0.965207	1.063990	-0.956689
4	1.036048	-0.939858	1.045272

Fig 8 Description: The above figure shows the output of the given code.

```
knn.fit(X_train,y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=1, p=2,
           weights='uniform')
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=1, p=2,
           weights='uniform')
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=1, p=2,
           weights='uniform')
```

Fig 9 Description: The above figure shows the output of the given code.

```
print(classification_report(y_test,pred))
```

	precision	recall	f1-score	support
0	0.95	0.68	0.79	565
1	0.73	0.96	0.83	520
micro avg	0.81	0.81	0.81	1085
macro avg	0.84	0.82	0.81	1085
weighted avg	0.84	0.81	0.81	1085

Fig 10 Description: The above figure shows the output of the given code.

```
plt.plot(range(1,600),error_rate,color='blue', linestyle='dashed', marker='o',markerfacecolor='red', markersize=10)
```

```
[<matplotlib.lines.Line2D at 0x22afbbdccf8>]
```

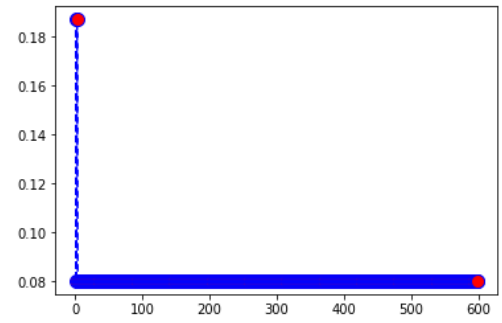

Fig 11 Description: The above figure shows the graph of the given code.

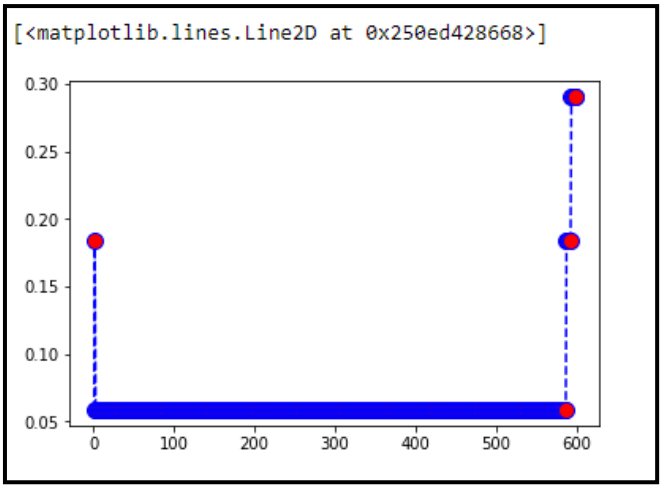

Fig 12 Description: Fig 3: The above figure represents the Error Rate vs K value of the KNN algorithm.

```
print(classification_report(y_test,pred))
```

	precision	recall	f1-score	support
0	0.95	0.68	0.79	565
1	0.73	0.96	0.83	520
micro avg	0.81	0.81	0.81	1085
macro avg	0.84	0.82	0.81	1085
weighted avg	0.84	0.81	0.81	1085

Fig 13 Description: The above figure shows the output of the given code.

```
print(classification_report(y_test,pred))
```

	precision	recall	f1-score	support
0	0.92	0.92	0.92	565
1	0.92	0.92	0.92	520
micro avg	0.92	0.92	0.92	1085
macro avg	0.92	0.92	0.92	1085
weighted avg	0.92	0.92	0.92	1085

Fig 14 Description: The above figure shows the output of the given code.

```
print(accuracy_score(y_test,pred))
```

```
0.9411764705882353
```

Fig 15 Description: The above figure represents the Accuracy Rate of the KNN algorithm.

```
print("Error rate is",1- accuracy_score(y_test, pred, normalize = True))
```

```
Error rate is 0.08018433179723505
```

Fig 16 Description: The above figure depicts the Error Rate of the KNN Algorithm.

```
X
```

```
array([[1, 0],
       [1, 1],
       [0, 1],
       ...,
       [0, 0],
       [1, 0],
       [1, 1]], dtype=int64)
```

```
Y
```

```
array([0, 1, 1, ..., 0, 0, 1], dtype=int64)
```

Fig 17 Description: The above figure shows the output of the given code.

```
# Fitting the classifier into the Training set


from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 20, criterion = 'entropy', random_state = 0)
classifier.fit(X_Train,Y_Train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=20, n_jobs=None,
            oob_score=False, random_state=0, verbose=0, warm_start=False)
```

Fig 18 Description: The above figure shows the output of the given code.

```
Y_Pred
```

```
array([1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1,
       1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0,
       0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1,
       1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0,
       0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
```

Fig 19 Description: The above figure shows the output of the given code.

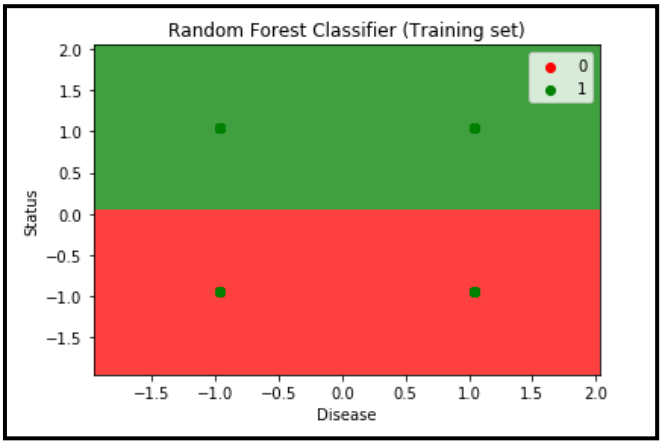

Fig 20 Description: The above figure visualizes the Training set results of the Random Forest algorithm.

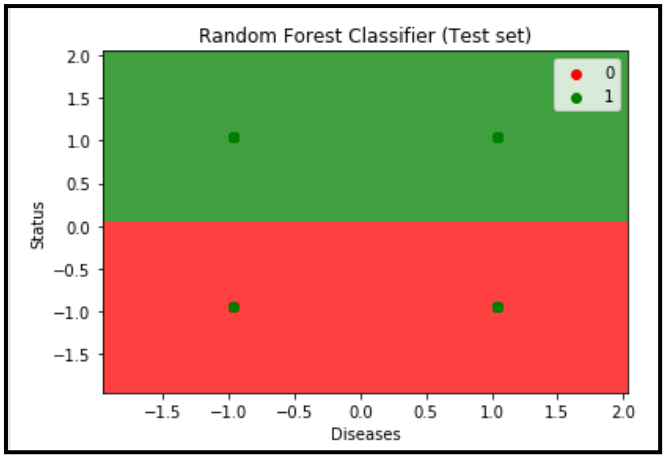

Fig 21 Description: The above figure visualizes the Testing set results of the Random Forest algorithm.

```
from sklearn import metrics

print('Mean Absolute Error:', metrics.mean_absolute_error(Y_Test, Y_Pred))
print('Mean Squared Error:', metrics.mean_squared_error(Y_Test, Y_Pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(Y_Test, Y_Pred)))
```

```
Mean Absolute Error: 0.0752212389380531
Mean Squared Error: 0.0752212389380531
Root Mean Squared Error: 0.27426490650109264
```

Fig 22 Description: The above figure depicts the MAE, MSE, RMSE values of the Random Forest Algorithm.

```
print(classification_report(Y_Test, Y_Pred))
```

	precision	recall	f1-score	support
0	0.91	0.93	0.92	480
1	0.92	0.90	0.91	424
micro avg	0.91	0.91	0.91	904
macro avg	0.91	0.91	0.91	904
weighted avg	0.91	0.91	0.91	904

Fig 23 Description: The above figure shows the output of the given code.

```
print(accuracy_score(Y_Test, Y_Pred))
```

```
0.9247787610619469
```

Fig 24 Description: The above figure depicts the Accuracy Rate of the Random Forest Algorithm.

```
print("Error rate is",1- accuracy_score(Y_Test, Y_Pred, normalize = True))
```

```
Error rate is 0.08517699115044253
```

Fig 25 Description: The above figure depicts the Error Rate of the Random Forest Algorithm.

```
df
```

	AGE	GENDER	LUNG DISEASE	HEART DISEASE	GASTRIC DISEASE	LOCALITY	STATUS	AFFECTED
0	51	F	1	0	0	CITY	DISEASED	0
1	45	M	1	1	1	CITY	DISEASED	1
2	21	F	0	1	0	CITY	DISEASED	1
3	56	M	0	1	0	VILLAGE	RECOVERED	1
4	41	F	1	0	1	CITY	DISEASED	0
5	15	M	1	0	1	CITY	DISEASED	0
6	12	M	0	1	0	VILLAGE	RECOVERED	1
7	51	F	0	1	1	VILLAGE	DISEASED	1
8	45	M	1	0	0	CITY	DISEASED	0
9	12	F	0	1	1	CITY	DISEASED	1
10	45	M	1	1	0	CITY	RECOVERED	1

Fig 26 Description: The above figure shows the output of the given code.

```
df_numeric_only
```

	AGE	LUNG DISEASE	HEART DISEASE	GASTRIC DISEASE	AFFECTED
0	51	1	0	0	0
1	45	1	1	1	1
2	21	0	1	0	1
3	56	0	1	0	1
4	41	1	0	1	0
5	15	1	0	1	0
6	12	0	1	0	1
7	51	0	1	1	1
8	45	1	0	0	0
9	12	0	1	1	1
10	45	1	1	0	1
11	84	1	0	1	0
12	45	0	1	0	1

3603	24	1	1	1	1
3604	70	0	1	1	1
3605	40	0	0	1	0
3606	43	1	0	0	0
3607	45	0	1	0	1
3608	49	0	0	1	0
3609	21	1	1	0	1
3610	47	1	0	1	0
3611	22	1	0	1	1
3612	68	0	1	1	1
3613	31	0	0	0	0
3614	53	1	0	0	0
3615	25	1	1	0	1

3616 rows × 5 columns

Fig 27 Description: The above figure shows the output of the given code.

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(2892, 4)
(724, 4)
(2892,)
(724,)
```

Fig 28 Description: The above figure shows the output of the given code.

```
y_pred
```

```
array([0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0,
       1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0,
       1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0,
       0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1,
       1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
```

Fig 29 Description: The above figure shows the output of the given code.

```
accuracy_score(y_test,y_pred)
```

```
0.925414364640884
```

Fig 30 Description: The above figure depicts the Accuracy Rate of the Naïve Bayes Algorithm.

```
print("Error rate is",1- accuracy_score(y_test, y_pred, normalize = True))
```

```
Error rate is 0.074585635359116
```

Fig 31 Description: The above figure depicts the Error Rate of the Naïve Bayes Algorithm.

```
df.tail()
df.shape
df.size
df.count()
```

```
AGE               3616
GENDER            3616
LUNG DISEASE      3616
HEART DISEASE     3616
GASTRIC DISEASE   3616
LOCALITY          3616
STATUS            3616
AFFECTED          3616
dtype: int64
```

Fig 32 Description: The above figure shows the output of the given code.

```
df
```

	AGE	GENDER	LUNG DISEASE	HEART DISEASE	GASTRIC DISEASE	LOCALITY	STATUS	AFFECTED
0	51	F	1	0	0	CITY	DISEASED	0
1	45	M	1	1	1	CITY	DISEASED	1
2	21	F	0	1	0	CITY	DISEASED	1
3	56	M	0	1	0	VILLAGE	RECOVERED	1
4	41	F	1	0	1	CITY	DISEASED	0
5	15	M	1	0	1	CITY	DISEASED	0
6	12	M	0	1	0	VILLAGE	RECOVERED	1
7	51	F	0	1	1	VILLAGE	DISEASED	1
8	45	M	1	0	0	CITY	DISEASED	0
9	12	F	0	1	1	CITY	DISEASED	1
10	45	M	1	1	0	CITY	RECOVERED	1
11	84	F	1	0	1	VILLAGE	RECOVERED	0
12	45	F	0	1	0	CITY	DISEASED	1
13	26	M	0	1	1	CITY	RECOVERED	1

Fig 33 Description: The above figure shows the output of the given code.

```
df_numeric_only
```

	AGE	LUNG DISEASE	HEART DISEASE	GASTRIC DISEASE	AFFECTED
0	51	1	0	0	0
1	45	1	1	1	1
2	21	0	1	0	1
3	56	0	1	0	1
4	41	1	0	1	0
5	15	1	0	1	0
6	12	0	1	0	1
7	51	0	1	1	1
8	45	1	0	0	0
9	12	0	1	1	1
10	45	1	1	0	1
11	84	1	0	1	0
12	45	0	1	0	1
13	26	0	1	1	1

Fig 34 Description: The above figure shows the output of the given code.

```
df.dtypes
```

```
AGE                int64
GENDER            object
LUNG DISEASE       int64
HEART DISEASE      int64
GASTRIC DISEASE    int64
LOCALITY          object
STATUS            object
AFFECTED           int64
dtype: object
```

Fig 35 Description: The above figure shows the output of the given code.

```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(2531, 4)
(2531,)
(1085, 4)
(1085,)
```

Fig 36 Description: The above figure shows the output of the given code.

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

           0       0.92      0.93      0.93       571
           1       0.93      0.91      0.92       514

   micro avg       0.92      0.92      0.92      1085
   macro avg       0.92      0.92      0.92      1085
weighted avg       0.92      0.92      0.92      1085
```

Fig 37 Description: The above figure shows the output of the given code.

```
print("accuracy of training dataset is{:.2f}".format(classifier.score(x_train,y_train)))
print("accuracy of test dataset is {:.2f}".format(classifier.score(x_test,y_test)))
```

```
accuracy of training dataset is0.92
accuracy of test dataset is 0.92
```

Fig 38 Description: The above figure depicts the Accuracy Rate of the SVM Algorithm.

```
print("Error rate is",1- accuracy_score(y_test, y_predict, normalize = True))
```

```
Error rate is 0.0755760368663595
```

Fig 39 Description: The above figure depicts the Error Rate of the Naïve Bayes Algorithm.

ALGORITHMS	ACCURACY	ERROR RATE
Decision Tree	96%	7.46%
K Nearest Neighbors	94%	8.01%
SVM	93%	7.57%
Naïve Bayes	92%	7.45%
Random Forest	92%	8.51%

Fig 40 Description: The above table illustrates the accuracy and error rate of different algorithms used for the Covid-19 prediction during experimentation.

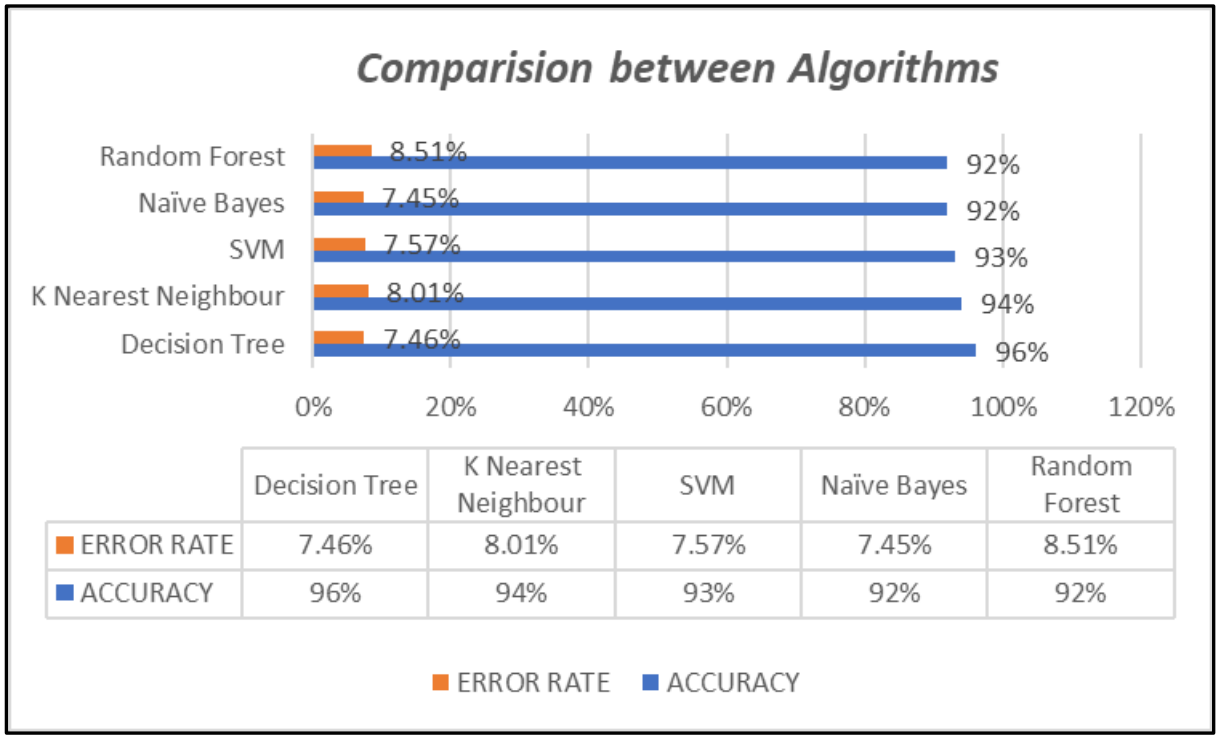

Fig 41 Description: The above figure illustrates the comparison between algorithms with the help of accuracy and error rate used for the Covid-19 prediction during experimentation.

12.CONCLUSION

In this paper, we predicted the COVID-19 cases by taking the data records of 3617 and by using the Decision Tree, SVM, Naïve Bayes, Random Forest, and KNN algorithms out of all the algorithms Decision Tree algorithm got the highest accuracy rate when compared with other algorithms.

13. FUTURE SCOPE

Currently, we predicted whether the person is cured or not cured due to COVID in future we can create models in such a way to predict the possible affected regions Based on the speed of increase of cases and the places that the affected people have visited and their geolocation we can be able to track what may be the affected areas in the future. And we can also use Hadoop and the concept of HDFS in such a way that we can be able to process huge datasets.

14. REFERENCES

[1] Chaurasia, V., & Pal, S. (2020). COVID-19 pandemic: ARIMA and regression model based worldwide death cases predictions.

[2] Tabik, S.,Gomez-Rios, A., Martin-Rodriguez, J. L., Sevillano-Garcia, I.,Rey-Area, M.,Charte, D., Guirado, E., Suarez, J. L., Luengo, J., Valero-Gonzalez, M. A., Garcia-Villanova, P., Olmedo-Sanchez, E., & Herrera, F. (2020). Covidgr dataset and COVID-sdnet methodology for predicting COVID-19 based on chest X-ray images. IEEE Journal of Biomedical and Health Informatics, 24(12), 3595-3605.

[3] Rochmawati, N.,Hidayati, H. B.,Yamasari, Y.,Yustanti, W.,Rakhmawati, L.,Tjahyaningtijas, H. P., & Anistyasari, Y. (2020). Covid symptom severity using decision tree. 2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE).

[4] Mandayam, A. U., A.C, R., Siddesha, S., & Niranjan, S. K. (2020). Prediction of COVID-19 pandemic based on regression. 2020 Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN).

[5] Ban, T. Q., Duong, P. L., Son, N. H., & Dinh, T. V. (2020). COVID-19 disease simulation using GAMA platform. 2020 International Conference on Computational Intelligence (ICCI).

[6] Nath, M. K., Kanhe, A., & Mishra, M. (2020). A novel deep learning approach for classification of COVID-19 images. 2020 IEEE 5th international conference on computing communication and automation (ICCCA).

[7] Gambhir, E., Jain, R., Gupta, A., & Tomer, U. (2020). Regression analysis of COVID-19 using machine learning algorithms. 2020 international conference on smart electronics and communication (ICOSEC).

[8] Fuzzy rule-based system for predicting daily case in COVID-19 outbreak. (2020). 2020 4th international symposium on multidisciplinary studies and innovative technologies (ISMSIT).

[9] Kumar, N., & Susan, S. (2020). COVID-19 pandemic prediction using time series forecasting models. 2020 11th international conference on computing, communication and networking technologies (ICCCNT).

[10] Bhadana, V., Jalal, A. S., & Pathak, P. (2020). A comparative study of machine learning models for COVID-19 prediction in India. 2020 IEEE 4th conference on information & communication technology (CICT).

[11] Latif, S., Usman, M., Manzoor, S., Iqbal, W., Qadir, J., Tyson, G., Castro, I., Razi, A., Boulos, M. N., Weller, A., & Crowcroft, J. (2020). Leveraging data science to combat COVID-19: A comprehensive review.

[12] Bhati, A., & Jagetiya, A. (2020). Prediction of COVID-19 outbreak in India adopting Bhilwara model of containment. *2020* 5th International Conference on Communication and Electronics Systems *(ICCES).*

[13] Singh, S., Raj, P., Kumar, R., & Chaujar, R. (2020). Prediction and forecast for COVID-19 outbreak in India based on enhanced epidemiological models. *2020* Second International Conference on Inventive Research in Computing Applications *(ICIRCA).*

[14] Istaiteh, O., Owais, T., Al-Madi, N., & Abu-Soud, S. (2020). Machine learning approaches for COVID-19 forecasting. 2020 International Conference on Intelligent Data Science Technologies and Applications (IDSTA).

[15] Alwaeli, Z. A., & Ibrahim, A. A. (2020). Predicting COVID-19 trajectory using machine learning. 2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT).

[16] Kumar, N., & Susan, S. (2020). COVID-19 pandemic prediction using time series forecasting models. 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT).

[17] Maurya, S., & Singh, S. (2020). Time series analysis of the COVID-19 datasets. 2020 IEEE International Conference for Innovation in Technology (INOCON).

[18] Laguarta, J., Hueto, F., & Subirana, B. (2020). COVID-19 artificial intelligence diagnosis using only cough recordings. IEEE Open Journal of Engineering in Medicine and Biology, 1, *275-281.*

[19] Liew, C., Quah, J., Goh, H. L., & Venkataraman, N. (2020). A chest radiography-based artificial intelligence deep-learning model to predict severe COVID-19 patient outcomes: The CAPE (COVID-19 AI predictive engine) model.

[20] Pham, Q., Nguyen, D. C., Huynh-The, T., Hwang, W., & Pathirana, P. N. (2020). Artificial intelligence (AI) and big data for coronavirus (COVID-19) pandemic: A survey on the state-of-the-Arts.

[21] Latif, S., Usman, M., Manzoor, S., Iqbal, W., Qadir, J., Tyson, G., Castro, I., Razi, A., Boulos, M. N., Weller, A., & Crowcroft, J. (2020). Leveraging data science to combat COVID-19: A comprehensive review.

PAPER.docx

ORIGINALITY REPORT

9%	5%	8%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	export.arxiv.org Internet Source	1%
2	doctorpenguin.com Internet Source	1%
3	Onder Tutsoy, Sule Colak, Adem Polat, Kemal Balikci. "A Novel Parametric Model for the Prediction and Analysis of the COVID-19 Casualties", IEEE Access, 2020 Publication	1%
4	Nagothu Srujan, Utlapalli Mahesh, Vemuri Poojitha, Syam Jason, Jonnalagadda Surya Kiran. "GuidoBot: A Novel Approach for Assisting Faculty in Tracking Goals and finding tasks", 2021 6th International Conference on Inventive Computation Technologies (ICICT), 2021 Publication	1%
5	"ICCI 2020 TOC", 2020 International Conference on Computational Intelligence (ICCI), 2020	1%

	Publication	
6	"Table of contents", 2020 IEEE 4th Conference on Information & Communication Technology (CICT), 2020 Publication	1%
7	"Table of contents", 2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE), 2020 Publication	1%
8	"ICCCA 2020 Subject Index Page", 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA), 2020 Publication	1%
9	"ICRCICN 2020 TOC", 2020 Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), 2020 Publication	1%
10	www.ismsitconf.org Internet Source	1%
11	docplayer.net Internet Source	<1%
12	ieeexplore.ieee.org Internet Source	<1%

A Novel Approach to Apply Different Algorithms to Predict Covid-19 Disease

[1] Utlapalli. Mahesh, [2] S. Nithya Tanvi Nishitha, [3] Bonela. Syam Jason, Jonnalagadda. Surya Kiran

[1,2] Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, A.P., India-522502.

[1] umahesh987654321@gmail.com , [2] Nithyatanvi9999@gmail.com, [3] syamjasonb777@gmail.com, [4] kiransurya93@kluniversity.in

Abstract: The Covid-19 episode happened in Wuhan in December 2019 and spread wherever on the planet. The Covid-19 transferable infection has an immunization and medication for its treatment. The premier significant components in lessening the spread of the infection are washing hands, utilizing a veil, and diminishing social distance. Today furthermore to clinical investigations, PC-supported examinations likewise are generally regulated for the Covid-19 episode. Computer-based intelligence strategies are effectively applied during this examination. In this investigation, we used different algorithms for the prediction and analysis of Covid-19 everyday cases. Because of the examination, the number of everyday cases was effectively assessed with these various kinds of algorithms.

Keywords: COVID-19, transferable infection, clinical investigations, computer-based intelligence, prediction.

I. INTRODUCTION

The essential case was distinguished in Wuhan, China, in December 2019, passing from bats to people. The infection spread everywhere in the world in a brief period and transformed into a lethal illness and fell the wellbeing frameworks of numerous nations in this world. Coronavirus is an extremely intense respiratory condition and infection movement can have deadly results. The clearest side effects of the illness; It is known as fever, dry hack, sore throat, migraine, shortcoming, muscle torment, looseness of the bowels, and windedness. In further developed cases, it causes serious pneumonia, irritating the lungs because of oxygen distinction and different organ disappointment. Particularly this infection has significantly more hazardous impacts for those with ongoing sicknesses, powerless obstruction or insusceptible framework, smokers, and the older. Man-made brainpower techniques [20] are utilized effectively in the arrangement of numerous issues. In the COVID-19 [11] scourge, numerous examinations have been completed utilizing computerized reasoning strategies. A portion of those are analysis utilizing radiology pictures, following sickness, assessing the patient's wellbeing result, early identification, and conclusion of disease, drug revelation, observing therapy.

II. LITERATURE SURVEY

A. Title: Covid Symptom Severity Using Decision Tree

Authors: Naim Rochmawati,Hanik Badriyah Hidayati, Yuni Yamasari ,Wiyli Yustanti ,Lusia Rakhmawati, Hapsari P. A. Tjahyaningtijas ,Yeni Anistyasari

In this paper, the main aim is to present a simple key whether it is symptoms of covid-19 [21] or normal cough. A clinical symptom dataset has been taken which is used for classifying the symptoms using the Decision Tree algorithm. In this research, the decision tree that has been used is hoeffding and j48. As we know that decision is one of the widely used algorithms for classification methods [12] as it is easy to simply by humans. the main aim is to cover the concept of concerting data into decision trees or decision rules. As a result, j48 has shown more efficiency than hoeffding tree in terms of recall, accuracy, precision. In further, this research work can be also used for the same dataset and different preprocessing. [1]

B. Title: COVIDGR Dataset and COVID-SDNet Methodology for Predicting COVID-19 Based on Chest X-Ray Images

Authors: S. Tabik , A. Gómez-Ríos, J. L. Martín-Rodríguez, I. Sevillano-García, M. Rey-Area, D. Charte, E. Guirado, J. L. Suárez, J. Luengo, M. A. Valero-González, P. García-Villanova, Olmedo-Sánchez, and F. Herrera

Current pandemic covid-19 is detected by taking the help of CT-Scans, X-Rays, CT scans. As we know that RT-PCR tests are mostly not available in medical centres and pharmacy stores hence CXR images have been used alternatively which is the most efficient, less cost, less time tool for assisting in taking the decisions. Here DL Neural networks have been used and shown potential for building [7] the covid-19 triage systems and to identify the covid-19 patients mostly detected patient with less severity. this research around two points i.e., clarified the hypersensitivity which is achieved in covid-19 classifications, well collaborate with most hospitals such as Spain, Granada. The dataset is stabilized by using the covidgr-1.0 that has every level. To use it in the future, it can enhance with covidgr-1.0 with a CXR image taken from multiple hospitals. [2]