PREDICTION AND CLASSIFICATION OF COVID-19 DEATH AND RECOVERED CASES USING MACHINE LEARNING

A major project report submitted in partial fulfilment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

Submitted by

Aditi Gupta (201529)

Shivangi Thakur (201288)

Under the guidance & supervision of

Mr. Prateek Thakral

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

CANDIDATE'S DECLARTION

We hereby declare that the work presented in this report entitled **'Prediction and classification of covid-19 death and recovered cases using machine learning'** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of **Mr. Prateek Thakral** (Assistant Professor (Grade-II), Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Aditi Gupta	Shivangi Thakur
201529	201288

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Mr. Prateek Thakral
Assistant Professor (Grade-II)
Computer Science and Engineering
Dated:

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the project work successfully.

I really grateful and wish my profound my indebtedness to Supervisor **Mr. Prateek Thakral Assistant Professor (Grade-II)**, Department of CSE Jaypee University of Information Technology, Waknaghat. Deep Knowledge & keen interest of my supervisor in the field of "**Machine Learning**" to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

I would like to express my heartiest gratitude to Prof. Dr. Vivek Kumar Sehgal**,** Head of Department of CSE/IT, for his kind help to finish my project.

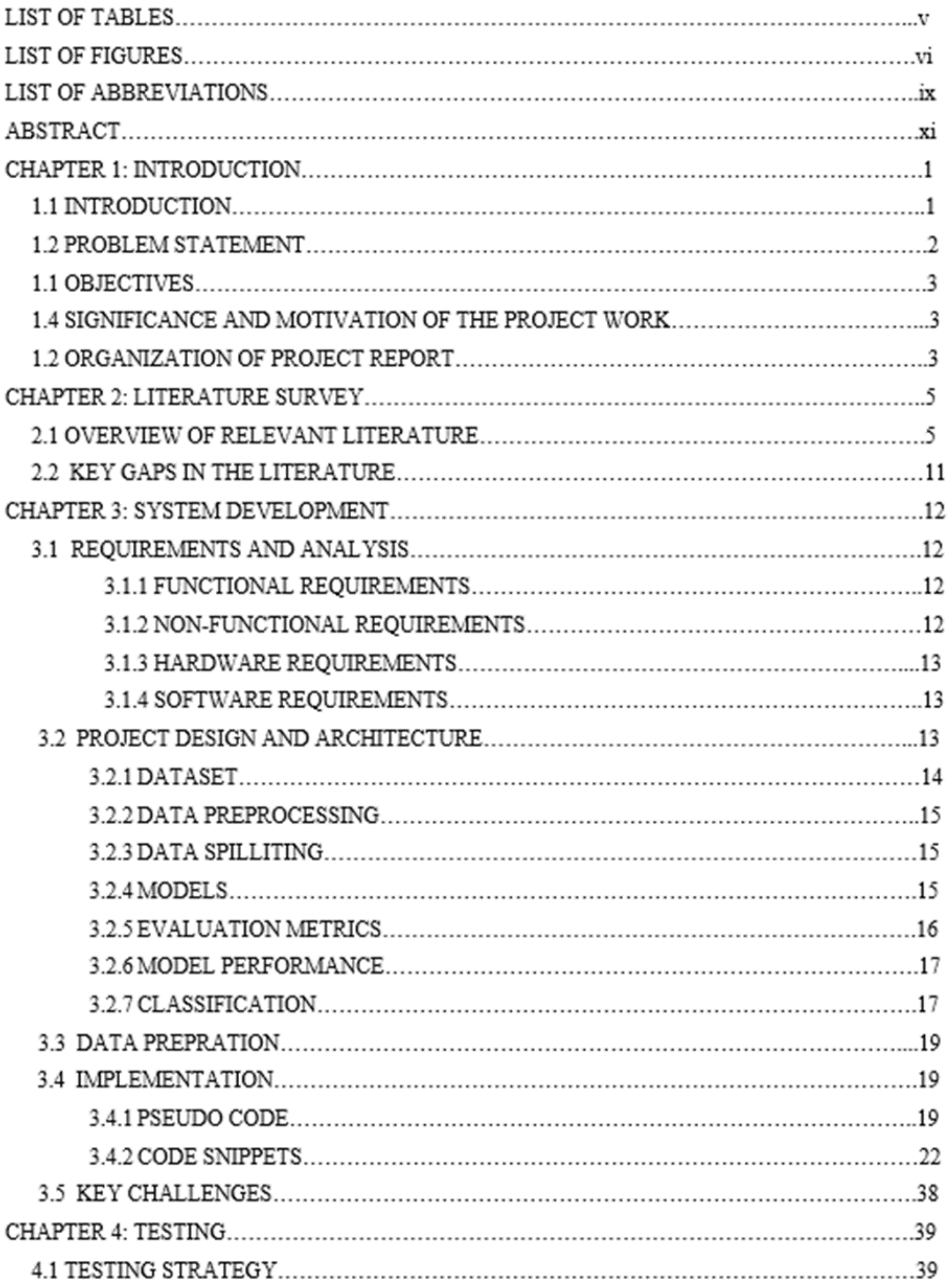
I would also generously welcome each one of those individuals who have helped me straight forwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Aditi Gupta
(201529)

Shivangi Thakur
(201288)

TABLE OF CONTENTS



LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Full Form
SVM	Support Vector Machine
XGB	Extreme Gradient Boosting
RF	Random Forest
KNN	K-Nearest Neighbors
GB	Gradient Boosting
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
LR	Linear Regression
PA	Prophet Algorithm
CNN	Convolutional Neural Network
ARIMA	Autoregressive Integrated Moving Average
BiLSTM	Bidirectional Long Short-Term Memory
GRUs	Gated Recurrent units
VAE	Variational autoencoder
MAPE	Mean absolute percentage error
LASSO	Least Absolute Shrinkage and Selection Operator
EV	External Validation
SSD	Solid State Drive
TPR	True Positive Rate
FPR	False Positive Rate
MCC	Matthew’s correlation coefficient
DFD	Data Flow Diagram
AUC	Area Under the Curve
ROC	Receiver Operating characteristic Curve
FP	False Positive
FN	False Negative

TP	True Positive
TN	True Negative
AWS	Amazon Web Services
CM	Confusion Matrix
SGD	Stochastic Gradient Descent

ABSTRACT

Over the years, the globe has faced serious epidemic and pandemic crises on a regular basis. Pandemics and epidemics have been a common occurrence for ages, continuing until the disastrous year 2020. With the advent of a new Coronavirus this year, the world has nearly come to a halt. The globe is currently under the Covid-19 umbrella following an outbreak because of its progression into a deadly pandemic. It has resulted in a global pandemic, leading to significant health and socio-economic challenges. As human being is unable to predict the covid-19 cases so there need to be system which can automatically predict and classify death and recovered cases. Today, fortunately we have advance machine learning and deep learning techniques which are used to construct frameworks which could predict and classify the cases. The effect on general health, encompassing the possibility of elevated infection, serious disease, and death rates. Public health is always at danger due to viral variations and difficulties containing its spread.

It has a profound psychological effect on individuals all throughout the world. The lockdowns and other measures that have been implemented to control the spread of the virus have caused widespread unemployment and business closures. The pandemic has also exacerbated existing social inequalities. Early identification of critical cases and accurate prediction of outcomes is crucial for effective resource allocation and patient management. In response to this unprecedented pandemic, researchers and healthcare professionals have been actively seeking innovative approaches to predict and classify COVID-19 cases, particularly in terms of patient outcomes.

Machine learning (ML) has emerged as a powerful tool for analyzing large datasets and making predictions, and it has the potential to be used for predicting COVID-19 outcomes. In response to this unprecedented pandemic, researchers and healthcare professionals have been actively seeking innovative approaches to predict and classify COVID-19 cases, particularly in terms of patient data. In outcome to this crisis, there is an analytical need for automated system that can predict and classify the cases. This project could be practically helping healthcare field for early Risk Identification and Intervention and Clinical Decision Support tool for healthcare professionals, and researchers and scientists by providing additional information to aid in clinical decision-making. We aim to is to harness the power of machine learning to make accurate predictions and classifications of COVID-19 death and recovered cases

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

Corona virus disease(covid-19) is an infectious disease caused by the SARS-CoV-2 virus. It has emerged as a global crisis with a profound impact on the economies, public health worldwide. SARS-COV-2 virus is the main reason of Corona virus disease. This virus belongs to the family of coronavirus, which is a category of virus that are known to cause this condition. In December 2019, the very first case of covid-19 was spot out in Wuhan, China. A defining characteristic of covid-19 is its high transmissibility, facilitated by respiratory droplets and close contact with the infected individuals.Within few days these symptoms may appear after the viral exposure. After infection most of the people have these variety of symptoms for years and also some of them found with permanent organ damage [4]. The illness rapidly spread globally led to covid-19 as a pandemic.

It transmits when infectious particle is inhaled or since it is a communicable disease spread more when come into touch or contact with the mouth, nose, or eyes. The risk is greater when people are close together, though minute airborne particles harbouring the virus can stay suspended in the air and move over longer distances, especially indoors To prevent it from spreading and increased number of researchers are attempting to develop different methods for predicting and classifying various covid cases. Variety of Machine learning algorithms can be implemented with large datasets of different persons clinical symptoms so, that it can be used for classification and discovering different patterns that make them apart.

It can be difficult in understanding and addressing the long-term health effects including persistent effects and complications. Therefore, it is crucial to create reliable algorithms that can effectively categorize and predict death and recovered cases while accounting from different clinical symptoms.

At this point of time there is a need to classify the death and recovered cases as soon as possible. The result of this project could aid in the effort to stop the increase in cases. This initiative can assist public health organizations in making more informed decisions and ultimately result in more informed and trustworthy decisions by deploying the model that classify the deaths and recovered cases accurately. Also, contributes to the scientific knowledge of covid-19 by identifying the key features and factors influencing this disease.

1.2 PROBLEM STATEMENT

The COVID-19 spread quickly and drastically and has posed an unprecedented global challenge, overwhelming healthcare systems and causing immense human suffering. India is one of the numerous nations in the globe that is currently experiencing COVID-19. The time-consuming nature of the current diagnosis processes to determine who is infected is impacting the diagnosis rate when handling a high volume of patients. As the pandemic continues, the ability to accurately predict the outcomes of cases and classify them into “Deaths” and “Recovered” categories has become crucial [5]. Accurate prediction and classification of COVID-19 outcomes, such as deaths and recoveries, are crucial for effective resource allocation, healthcare planning, and policy making. Predicting the outcomes of COVID-19 cases (i.e., whether a patient will recover or succumb to the disease) and categorizing patients into these outcome groups based on their features are the two main components of the current challenge.

Today, fortunately we have advance machine learning and instruments which offer incredible guarantee for specialists to construct frameworks which could predict the cases. It identifies significant features and patterns, learns from historical data, and creates models for prediction. These models offer real-time decision support, aid resource allocation, and provide insights for healthcare professionals and policymakers. Through continuous learning and adaptation, machine learning enhances accuracy and contributes valuable insights to effectively manage the pandemic. This project has significant applications across healthcare, public health, research, data science, and emergency response fields. Support tool for healthcare practitioners, providing additional information to aid in clinical decision-making and aim to harness the power of machine learning to make accurate predictions and classifications of COVID-19 death and recovered cases. The healthcare professionals have been actively seeking innovative approaches for predicting and classifying COVID-19 cases, particularly in form of patient outcomes. To accurately classify COVID-19 cases into two categories i.e. Death and Recovered cases.

So, by developing a comprehensive, adaptable, and ethically sound model. The project's outcomes hold the potential to revolutionize patient care, resource allocation, and public health strategies, contributing valuable insights to the medical community's understanding of COVID-19 outcomes.

1.3 OBJECTIVES

The objectives of the project are:

- To Develop a covid-19 prediction system.
- To utilize machine learning algorithms to automate the prediction and classification of covid-19 death and recovered cases.
- To identify patterns and trends in covid-19, in order to better understand how and why it is spread.

1.4 SIGNIFICANCE AND MOTIVATION OF THE PROJECT WORK

Covid-19 effects the world badly and at that time there would be high need for such models that classify and predicts the covid-19 cases accurately. The cases must be predicted accurately as he/she was found positive or negative. There must be a classifier that classify the number of recovered and death cases respectively. As number of cases increases there is a need of accurate and time decision support tool. So, this task to forecast the cases is solved by machine learning. By developing accurate classification models, this project seeks to provide healthcare professionals with a proactive tool for identifying individuals at a risk of adverse outcomes. Motivation for this project is the urgent need for advanced tools to assist healthcare systems in navigating the complexities of the ongoing global pandemic.

1.5 ORGANISATION OF THE PROJECT REPORT

We organized our project into various steps:

- Introduction

  The virus SARS-CoV-2 belongs to the family of coronavirus, which is a category of virus that are known to cause this condition. It had various symptoms to identify the covid such as cough, fever, headache, fatigue, beathing problem and most important loss of flavour and aroma. Within one to fourteen days symptoms started to appear after the viral exposure. To preclude it from spreading and increased number of researchers are attempting to develop different methods that help in predicting and classifying the number of covid-19 cases.
- Literature Survey

Analysis of COVID-19 Death Cases Using Machine Learning, Time series predicting of COVID-19 based on deep learning, Prediction of COVID-19 Confirmed, Death, and Cured Cases in India Using Random Forest Model, and many other research articles were found among we reviewed. So as to classify and predict of covid-19 cases many machine learning and deep learning techniques were used.

- System Development

  The requirement analysis is done with respect to the function, non-functional requirements, hardware, and software requirements etc. Then the project architecture is described with the figures and the other figure describes the data flow diagram with respect to three levels. After this data preparation is done from different sources the implementation part is done where different models are applied and accuracy is calculated of test data. At last, the key challenges that can while implementation are described.

- Testing

  Different testing strategies were applied such as cross validation, train test split, confusion matrix and many others so that the models can form accurately.
- Results and evaluation

  In this different graph were plotted and different models are compared with respect to cross validation and without cross validation.
- Conclusion and future scope

  The project came to end and where we concluded with the models and what we would do next in the project.

CHAPTER 2: LITERATURE SURVEY

2.1 OVERVIEW OF RELEVANT LITERATURE

Several machine learning and deep learning techniques are used for predicting and classifying covid-19 cases have been proposed in the literature. According to published research many approaches and features like medical imagining, clinical features, laboratory test results, and deep learning models, machine learning models can increase the prediction and efficacy by providing valuable insights for better recognition of covid-19. The several improvements made by researchers to the topic of predicting and classifying covid-19 cases are analyzed in this section.

H. Aslam and S. Biswas [6] presented a method for real-time forecast for death cases of covid-19. They used base classifiers as XGBoost (XGB), Support Vector Machine (SVM) and Random Forest (RF). The Our World in Dataset of COVID-19 is used for evaluation purpose. They also generated a real-time forecast for the death cases of covid-19 using the techniques. There are fifteen possible features and one is the new death per million. Also, identified that ten features are giving higher impact on the covid-19 death cases. They used performance matrix such as RMSE, MAE to measure the model's performance. The experimental results give some suggestions and insights about reducing the mortality rate of covid-19 and shows that Support Vector Machine and XGBoost classifiers achieve the best performance.

M.O Alassafi et al. [7] suggested the use of some deep learning models to predict the spread of covid-19 in Malaysia, Morocco and Saudi Arabia. The dataset is taken from European Centre for Disease Prevention and Control. This approach compared two models of deep learning such as, Long Short-Term Memory (LSTM) and Recurrent Neural Network (RNN) to predict the number of covid-19 cases. They also predicted confirmed death cases for seven days using the available data. The results shows that Long Short-Term Memory performs better than Recurrent Neural Network in predicting the number of covid-19 deaths cases with a precision accuracy of 98.58% and reduce the error value. As RNN model was unstable and prone to vanishing gradient error and also in future they plan to apply machine learning.

R. Kumari et al. [8] employed an approach for analyzing and forecasting the figure of confirmed, death and recovered, cases in India of covid-19 using various statistical models. The dataset is collected from December 2019 to June 2020 to analyze the impact of lockdown on the spread of the disease. They presented different base classifiers such as Correlation Coefficient, Multiple Linear Regression, autocorrelation and, autoregression for prediction of the cases. They also suggested that prediction could be helpful for implementing preventive measures to control the spread. The predicted value of R-squared is 0. 9992.Also, predicted that the actual and predicted value are in good agreement with mean absolute error of less than 10%.

V.K. Gupta et al. [9] presented a solution for detecting and forecasting covid-19 cases in different states of India in chronological dates. The multiclass classification is used for the prediction. They presented a framework for detecting covid-19 cases that consists of 4 stages: (1) data cleaning, (2) feature selection, (3) performed forecasting, (4) measure consistency of the model. The bases classifiers used are RF, LR, SVM, and Neural Network are used to train the model based on accuracy. They also applied K-fold cross validation to measure the consistency. The experimental results show that Random Forest has the highest accuracy of 83.54%.

M. Alazab et al. [10] provided a strategy for detecting and predicting covid-19 patients using deep learning models such as ARIMA, PA, LSTM, for detection of chest X-rays images CNN is used and for diagnosis they used VGG16 model. They used datasets such as X-ray images to detect covid-19. The experimental results revel that Prophet Algorithm delivers best performance of 99.94% and VGG16 achieves an F-measure of 99% with augmented dataset and also analyzed the spatial distribution of covid-19 cases and these can help medical staff to deal with the covid-19 pandemic effectively and finds that the coastal areas are most affected by the disease than other areas respectively.

A. Zeroual et al. [11] used many deep learning methods to forecast the covid-19 cases by using RNN, LSTM, BiLSTM, GRUs, and VAE. The dataset Johns Hopkins University is used in which six different countries data is collected for forecasting. They used a performance verification technique such as EV, MAE, RMSE, MAPE, and RMSLE matrix. The proposed approach suggested that Variational Autoencoders archives enhanced forecasting compared to all other models and this can help government and decision makers on the issues and the necessary remedies at that time.

P. Wang et al. [12] discussed various approaches used for predicting covid-19 cases using machine learning technics and logistic model. The data is epidemiological data collected before June 16, 2020 is used to determine the pandemic trend. They also proved that the number of cases verified is equal to the number of cumulative verified cases and less than the number of reclaimed and death cases. The majority of the reviewed research used the FbProphet model as it is based on time series projections .so, to calculated the epidemic curve and project the trends of covid-19 this can improve the estimates of infection and effectiveness in computing globally results.

V. Bhadana et al. [13] proposed a relative study of various models of machine learning for forecasting and decision making. The base classifiers are LR, DT, RF and Least Absolute Shrinkage, SVM and LASSO for forecasting covid-19 active cases, death and recovered cases in next five days. They also presented a six-degree polynomial experiment and poly LR and ploy LASSO provides best results and SVM provides very poor results because of dataset fluctuations and it is challenging to divide the values by hyperplane and Random Forest and Decision Tree provides overfitting.

C. An et al. [14] uses different techniques of machine learning to develop the model that predicts the forecasting of Covid19 patients based on sociodemographic information, transmission pathway for the whole of South Korea. They use LASSO and linear SVM, Random Forest, methods for the prediction of Covid-19. Among all of them LASSO and linear SVM give the best results with high sensitivities and >90% of their accuracies. Machine learning focuses on high prediction accuracy than how accuracy is achieved. The primary problem of this research is that data does not contain the information regarding laboratory or radiologic. They created the model with the goal at early forecasting and prediction at the time of diagnosis prior to additional diagnostic therapy. At last, they created and verified machine learning models to predict the covid19 patients. LASSO and Linear SVM give best results with highest sensitivity and specificity for detecting individuals at risk.

S. Shastri et al. [15] suggested a deep learning framework that predicts the covid 19 confirmed and fatality cases in India and united states. They use RNN and LSTM variants such as stacked, Bi-directional and convolutional LSTM to design proposed models. Among them only Convolution LSTM gives better performance than other two models and provides results with high accuracy with less error. LSTM is a modified RNN that is used to overcome the RNN limitations. In the near future, we can examine the entire economic loss in the completion of Covid-19 in different areas and make a effective recovery strategy which can assist the countries recovery and their economy rate. We desire to foresee possible Covid-19 instances in other countries and the aerosol transmission of Covid-19 can be proven. The best model is chosen based on the error rate in predictions and the error is calculated using Mean Absolute Percentage Error

(MAPE). The convolutional LSTM model outperforms the other two models, with error rates ranging from 2.0 to 3.3 percent across all four datasets.\
S.S Aljameel et al. [16] focuses on the prediction of prognosis of covid19 patients based on the features or symptoms recognised during the quarantine period. Their data contains 287 samples from King Fahad University Hospital in Saudi Arabia. They used three machine learning algorithms that are LR, RF, and XGB. For the next phase that is preprocessing, they used data balancing using SMOTE. Their data contain 12 features that help to predict the analysis of survival and death of patients. They found among all methods only Random Forest outperforms the other methods with highest accuracy of 0.95 and AUC of 0.99 with 10-fold cross validation. The limitation of this is that there is a need for further validation with multiple datasets and future exploration by adding new features in the dataset.
E. Fayyoumi et al. [17] presents an approach using machine learning to recognize potential coronavirus patients and to stop or control the spread of covid19. Several machine learning and statistical models including LR, SVM and MLP are used to predict the cases depending on their signs or symptoms. They came to the conclusion that among all models only MLP Model achieves the highest accuracy of 91.62%. Also conclude that SVM gives best precision of 91.67%. Their future work is firstly to implement these models in Jordanian hospitals to identify covid19 patients quickly and safely and hence helpful in reducing the rapid spread of covid19 virus. They planned to implement these models in the large dataset by exploring their parameters in depth for further improvement and effectiveness.
The aforementioned studies indicate the following (1) In general, we examined various predictions and classification of death and recovered cases on various datasets such as clinical symptoms, country based etc. (2) A better approach is need to combine various models to increase the accuracy and other factors. (3) researchers have yet to explore tools and technologies which can classify and predict more accurate results. Hence, to resolve the prediction and classification problem we have implemented certain models to provide more accurate and efficient results in our project.

Table 2.1 Literature Survey

S. No.	Paper Title [Cite]	Journal/Conference (Year)	Tools/Techniques/ Dataset	Results	Limitations
1.	Analysis of COVID-19 Death Cases Using Machine Learning. [2]	SNComputer Science/ Springer (2023)	XGBoost, RF and SVM. Dataset-Our World in Data COVID-19.	The SVM and XGBoost give accurate results.	Many missing values in original data set.
2.	Time series predicting of COVID-19 based on deep learning.[3]	Neurocomputig /Science direct (2022)	RNN and LSTM deep-learning networks. Dataset- European Centre	The LSTM Precision and accuracy are 98.58% while the RNN has 93.45%	Few models are used for the prediction of Covid-19 pandemic.
3.	Analysis and Predictions of Spread, Recovery, and Death Caused by COVID-19 in India.[4]	Big Data Mining and Analytics, IEEE Xplore (2021)	Correlation coefficient, Multiple linear regression, autocorrelation , autoregression.	Auto correlation shows a good agreement with 0.9992 R-squared score.	ML Regression and autoregression were not used to predict the cases.
4.	Prediction of COVID-19 Confirmed, Death, and Cured Cases in India Using Random Forest Model.[5]	Big Data Mining and Analytics, IEEE Xplore (2021) Department of Computer Scie	Random forest, LR, SVM, decision tree, Dataset- Ministry of Health & Family Welfare.	Random forest has the highest accuracy of 83.54%, 72.79%, and 81.27%.	Semi-supervised hybrid design to identify COVID-19.

5.	Machine Learning-Based Model to Predict the Disease Severity and Outcome in COVID-19 Patients [6]	Hindawi Scientific Programming (2021)	logistic regression (LR), random forest (RF), and extreme gradient boosting (XGB)	RF outperformed the other classifiers with an accuracy of 0.95 and area under curve (AUC) of 0.99	Models need to be validated using multiple datasets.
6.	Deep learning methods for forecasting COVID-19 time-series data: A Comparative Study.[7]	Chaos,solitons & fractals/ Elsevier (2020)	RNN, LSTM, BiLSTM,GRUs, VAE. Dataset-Johns Hopkins University CSSE	VAE achieved better forecasting performance	Limited dataset which may not be representative of the global situation.
7.	Prediction of epidemic trends in COVID-19 with logistic model and, machine learning technics.[8]	Chaos,Solitons & Fractals/ Elsevier (2020)	Logistic growth forecasting model, FbProphet model Dataset-John Hopkins University	FbProphet model is a powerful tool for time series forecasting based on the data	It is based on assumption and input cases and some countries are not taken into account.
8.	A Comparative Study of Machine Learning Models for COVID-19	Conferences/ IEEE 4th Conference on Information & Communication Technology (CICT) (2020).	LASSO, Random Forest, Decision tree, LR, SVM, polynomial regression.	Poly LR and poly LASSO gives the best results.	SVM shows the poor result in the prediction of COVID-19.

	prediction in India.[9]		Dataset-Api.covid19india.org		
9.	COVID-19 Prediction and Detection Using Deep Learning.[10]	Soft computing (2020)	ARIMA, PA, LSTM, VGG16 Dataset-COVID-19 case data from Australia and Jordan	PA algorithm Gives an accuracy of 99.94% and VGG16 achieve 99%	Unable to compare the performance using large dataset
10.	Machine Learning and Statistical Modelling for Prediction of Novel COVID-19 Patients Case Study: Jordan [11]	International Journal of Advanced Computer Science and Applications	(Logistic Regression, LR) and (Support Vector Machine, SVM, and Multi-Layer Perceptron, MLP)	The MLP has shown the best accuracy (91.62%) compared to the other models and SVM gives precision 91.67%.	Not able to apply these models on big dataset.

2.2 KEY GAPS IN THE LITERATURE

The provided literature survey has some key gaps such as very few models are applied for the prediction. The papers do not use linear regression and autoregression to foresee the Covid-19 cases, which could improve some accuracy and reliability of the results. It also does not employ some of the data features that could be relevant for the prediction, such as missing values, input cases, and spatial influence of countries. It relies on a limited dataset that may not reflect the global situation of the pandemic, some models have poor performance in the prediction of COVID-19, which could indicate a need for better parameter tuning or model selection, increase the dataset so that it can quickly helpful in reducing the rapid spread of covid19 virus.

CHAPTER 3: SYSTEM DEVELOPMENT

3.1 REQUIREMENTS AND ANALYSIS

The requirement and analysis are mentioned below.

3.1.1 FUNCTIONAL REQUIREMENTS

A system's functional requirement specifies what the system should be able to achieve. These are the product features that user needs and indicates functioning of software system. These are as follows –

- Data collection: The system must be capable of gathering different datasets of covid-19 and integrating from diverse source. which contains the number of death and recovered cases and ensure accuracy and handling the variation.
- Data preprocessing: The system must have the capability to preprocess the gathered data by dropping the values that are inconsistent, handling missing values, standardizing variables, drop duplicates, and ensuring data quality, or using other methods of feature engineering.
- Feature extraction: The System must be able to extract useful aspects from the pre-processed data, including the useful columns having more impact on the dataset.
- Model evaluation: The system must be able to assess the effectiveness of the machine learning models using appropriate evaluation like recall, precision, F1-score, accuracy.
- Real- time updates – The system must support real time data updates to keep pace with dynamic nature.

3.1.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements are those that are not directly related to the system's declared functions. These are as follows –

- Maintainability: The system should be easy to maintain, with clear documentation and support for troubleshooting and bug-fixing.
- Security: The system needs to be secure. In order to prevent unauthorized access and ensure that the data it processes is not altered.

- Usability: The system should be user-friendly, with easy-to-use interface that allows users to input data and view results easily.
- Scalability: The system should be able to handle large amounts of data. It should be able to scale up to handle larger datasets and models as the project grows.
- Compatibility: The system should be compatible with different devices, software's, operating systems, and browsers.

3.1.3 HARDWARE REQUIREMENTS

- Minimum 8 GB of RAM.
- Solid-state drive (SSD) for faster read and write speeds.
- Processor intel/amd.
- Storage of 128GB.

3.1.4 SOFTWARE REQUIREMENTS

- Python.
- Google Collab.
- Machine Learning Frameworks.
- Streamlit.

3.2 PROJECT DESIGN AND ARCHITECTURE

The proposed architecture in Fig. 3.1 consists different steps that demonstrates the architecture of the project. Here, each of these steps are future discussed.

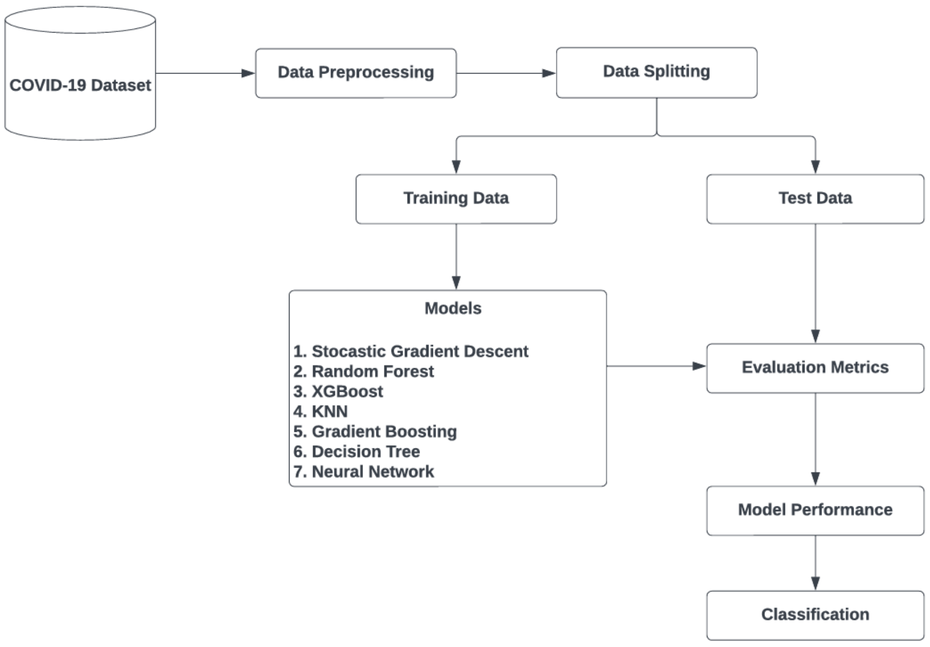


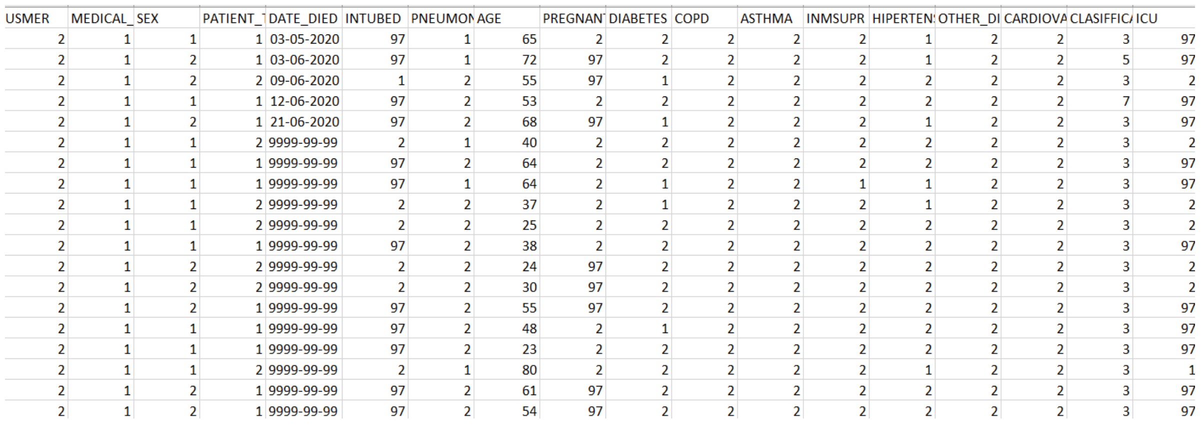
Fig 3.1: Visual representation of architecture

3.2.1 DATASET

It is evaluated on the clinical symptoms of the person from different hospitals and merged together.

- Before Feature Selection

  In this we have 18 features and 40000 rows. The figure 3.2 describes first 20 rows of the dataset.

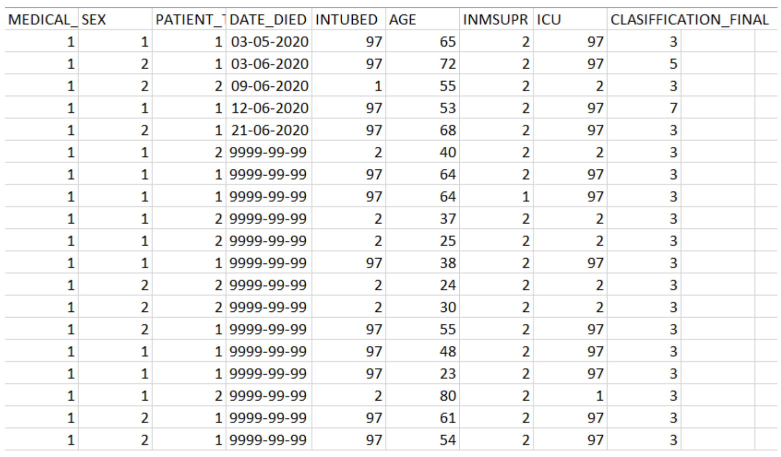


USMER	MEDICAL_	SEX	PATIENT_	DATE_DIED	INTUBED	PNEUMON	AGE	PREGNANT	DIABETES	COPD	ASTHMA	INMSUPR	HIPERTEN:	OTHER_DI	CARDIOVA	CLASIFFIC/	ICU
2	1	1	1	03-05-2020	97	1	65	2	2	2	2	2	1	2	2	3	97
2	1	2	1	03-06-2020	97	1	72	97	2	2	2	2	1	2	2	5	97
2	1	2	2	09-06-2020	1	2	55	97	1	2	2	2	2	2	2	3	2
2	1	1	1	12-06-2020	97	2	53	2	2	2	2	2	2	2	2	7	97
2	1	2	1	21-06-2020	97	2	68	97	1	2	2	2	1	2	2	3	97
2	1	1	2	9999-99-99	2	1	40	2	2	2	2	2	2	2	2	3	2
2	1	1	1	9999-99-99	97	2	64	2	2	2	2	2	2	2	2	3	97
2	1	1	1	9999-99-99	97	1	64	2	1	2	2	1	1	2	2	3	97
2	1	1	2	9999-99-99	2	2	37	2	1	2	2	2	1	2	2	3	2
2	1	1	2	9999-99-99	2	2	25	2	2	2	2	2	2	2	2	3	2
2	1	1	1	9999-99-99	97	2	38	2	2	2	2	2	2	2	2	3	97
2	1	2	2	9999-99-99	2	2	24	97	2	2	2	2	2	2	2	3	2
2	1	2	2	9999-99-99	2	2	30	97	2	2	2	2	2	2	2	3	2
2	1	2	1	9999-99-99	97	2	55	97	2	2	2	2	2	2	2	3	97
2	1	1	1	9999-99-99	97	2	48	2	1	2	2	2	2	2	2	3	97
2	1	1	1	9999-99-99	97	2	23	2	2	2	2	2	2	2	2	3	97
2	1	1	2	9999-99-99	2	1	80	2	2	2	2	2	1	2	2	3	1
2	1	2	1	9999-99-99	97	2	61	97	2	2	2	2	2	2	2	3	97
2	1	2	1	9999-99-99	97	2	54	97	2	2	2	2	2	2	2	3	97

Fig 3.2 Screenshot of dataset before feature selection

- After Feature

  In this we have used corelation coefficient for feature selection and a threshold for the feature selection ad left with 8 features and 22592 rows. Figure 3.3 describes first 20 rows of the dataset.



MEDICAL_	SEX	PATIENT_T	DATE_DIED	INTUBED	AGE	INMSUPR	ICU	CLASIFFICATION_FINAL
1	1	1	03-05-2020	97	65	2	97	3
1	2	1	03-06-2020	97	72	2	97	5
1	2	2	09-06-2020	1	55	2	2	3
1	1	1	12-06-2020	97	53	2	97	7
1	2	1	21-06-2020	97	68	2	97	3
1	1	2	9999-99-99	2	40	2	2	3
1	1	1	9999-99-99	97	64	2	97	3
1	1	1	9999-99-99	97	64	1	97	3
1	1	2	9999-99-99	2	37	2	2	3
1	1	2	9999-99-99	2	25	2	2	3
1	1	1	9999-99-99	97	38	2	97	3
1	2	2	9999-99-99	2	24	2	2	3
1	2	2	9999-99-99	2	30	2	2	3
1	2	1	9999-99-99	97	55	2	97	3
1	1	1	9999-99-99	97	48	2	97	3
1	1	1	9999-99-99	97	23	2	97	3
1	1	2	9999-99-99	2	80	2	1	3
1	2	1	9999-99-99	97	61	2	97	3
1	2	1	9999-99-99	97	54	2	97	3

Fig 3.3 Screenshot of dataset after feature selection

3.2.2 DATA PREPROCESSING

This step includes data filtering, transforming, data labelling, dropping unwanted columns, removing unwanted columns, and doing feature selection additionally performed data cleaning to remove irrelevant rows and consistency. In the dataset we have 8 dependent variables and one independent variable.

3.2.3 DATA SPLITTING

Divided the data into two sub sets as training set and testing set in the ratio of 80%-20% using hold out process.

3.2.4 MODELS

- Stochastic Gradient Descent – It is an iterative method used in machine learning during each search a random weight is selected refers to a few basic iterative structures thar are used to solve root-finding and stochastic optimisation issues. It runs one training example per iteration. It is also known as stochastic approximation [18].

- Random Forest – It is an ensemble learning approach. It combines the results of several decision trees to get a single result [19]. Compared to individual decision trees, this approach is more resistant to overfitting and frequently achieves superior accuracy on real world datasets.
- XGBoost – Extreme Gradient Boosting is a popular machine learning algorithm that belongs to ensemble learning methods. It is the implementation of gradient boosting decision tree It is particularly known for effectiveness and various tasks such as classification, regression etc. [20]. It provides regularization which helps to control overfitting by introducing L1/L2.
- KNN – K-Nearest Neighbour is a supervised machine learning algorithm. It is non-parametric algorithm and also a lazy learner algorithm [21]. It stores all the available data and classifies new data point based on the similarity between the data.
- Gradient Boosting- It is an ensemble learning method which is used for building predictive models. It is used both in classification and regression tasks. It is a functional gradient algorithm that repeatedly selects a function that leads in the direction of weak hypothesis so that it can minimize the loss function [22].
- Decision Tree – It is a type of supervised machine learning algorithm. It is used for both classification and regression. It creates a flow chart like tree structure in which each internal node represents a test on an attribute, each branch reflects the result of the test and each leaf node has a class label [23].
- Neural Network – It is machine learning model that makes judgement in a manner comparable to human brain, by using techniques that mirror the way biological neurons interacts to discover events and arrive at a conclusion [24].

3.2.5 EVALUATION METRICS

Different types of evaluation metrics are as follows –

- TPR – Also called recall, it measures how many cases are accurately predicted.
- FPR – It is the percentage of cases that gets misidentified.
- Precision – It is the degree of exactness.
- F1 Score – It is defined as the harmonic mean of recall and precision.
- Accuracy (%) – It is the portion of cases that are classified successfully.

- MCC – It is employed to assess how well machine algorithm performs in binary classification. It takes values between -1 and +1, measuring the correlation between the actual and expected labels [23].

3.2.6 MODEL PERFORMANCE

To increase model's performance different techniques are used such as -

- Stacking – It integrates the prediction of different base classifiers into a meta model in order to get more extraordinary performance than utilizing a single model.
- Hyperparameter tunning – It is the process of selecting the best values for a machine learning model hyperparameters. It is configuration that regulates the models learning process.

3.2.7 CLASSIFICATION

It means classifying the case in death or recovered using a system which is made using streamlit it is an open-source framework that makes the dynamic apps with only few lines of code.

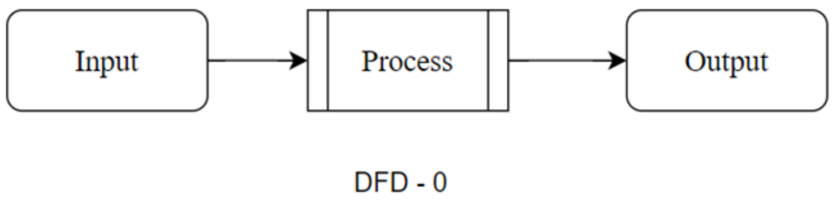


Fig 3.4 Depicts DFD level 0

DFD - 0 firstly we give input of the symptoms to the system for the examination. Then, the system performs the processing of the symptoms and extract the relevant features.

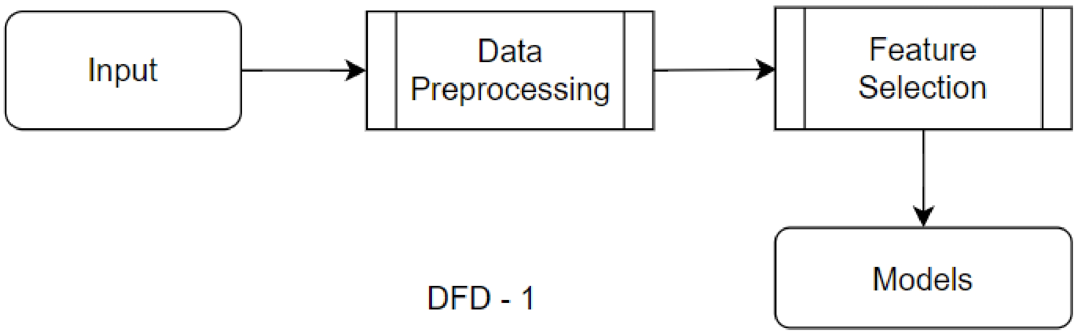


Fig. 3.5 Depicts DFD level 1

DFD – 1 provides more detailed and structured processes as compared to the level 0 DFD. In this after data preprocessing, we do feature selection to extract and select all the relevant features. Then all the relevant models are applied for the classification purpose.

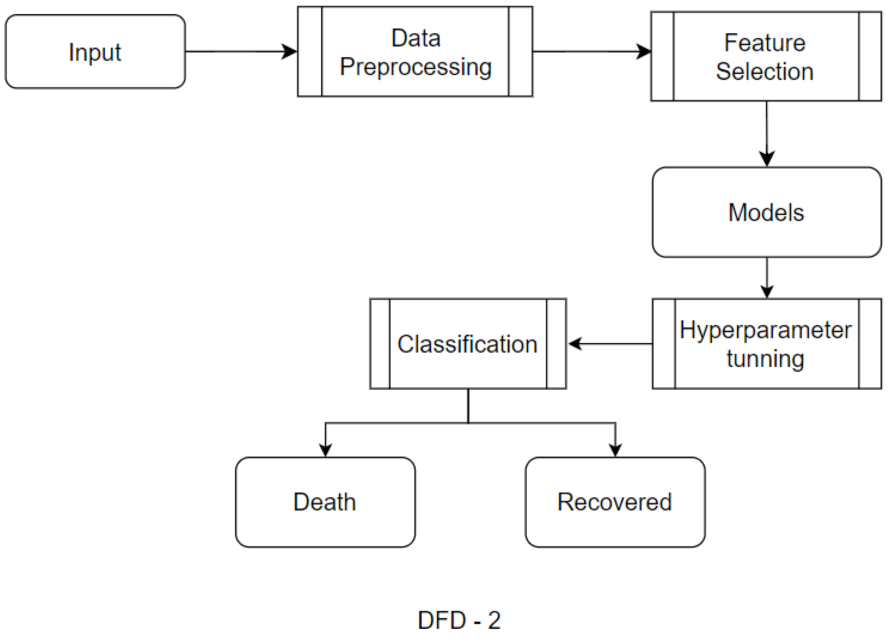


Fig. 3.6 Depicts DFD level 2

DFD – 2 level provides more detailed view of all the processes involved in DFD level 1 into more detailed processes. After the data is provided, preprocessing, cleaning and feature selection are done and the machine learning models are applied then classification is done for the death and recovered cases.

3.3 DATA PREPARATION

The data preparation was an essential and prolonged process. As clinical information of patients is not presented publicly so from various hospitals and health institutions patient's data was collected and merged into one meta data file stored in csv format. The data consists of 40,000 number of rows and 18 number of columns from that columns only 8 columns are select for the training and testing part of the models such as medical unit, sex, patient type, ventilator, age, inmsupr, icu, positive.

3.4 IMPLEMENTATION

It describes the main pseudo code and code snippets as follow –

3.4.1 PSEUDO CODE

1. Pseudo code of Stochastic Gradient Descent

```
function SGD (covid-19_dataset):
#Split the dataset into training and testing sets
train_set, test_set = split_dataset(covid-19_datset)
# Extract the features from the training set
features = extract_feature(train_set, features)
#Train the SGD classifier
sgd_classifer = train_sgd(train_set,features)
# Evaluate the sgd classifier for testing
Accuracy = evaluate_sgd(sgd_classifier, test_set)
# Returns the accuracy and sgd classifier
Return sgd_classifier,accuracy
```

2. Pseudo code of Random Forest

```
function RandomForest (covid-19_dataset):
#Split the dataset into training and testing sets
train_set, test_set = split_dataset(covid-19_datset)
# Extract the features from the training set
```

```
features = extract_feature(train_set, features)
#Train the Random Forest classifier
rf_classifer = train_random_forest(train_set,features)
# Evaluate the Random Forest classifier for testing
Accuracy = evaluate_random_forest(rf_classifier, test_set)
# Returns the accuracy and Random Forest classifier
Return rf_classifier,accuracy
```

3. Pseudo code of XGBoost

```
function XGBoost (covid-19_dataset):
#Split the dataset into training and testing sets
train_set, test_set = split_dataset(covid-19_datset)
# Extract the features from the training set
features = extract_feature(train_set, features)
#Train the XGBoost classifier
xg_classifer =  train_xgboost(train_set,features)
# Evaluate the XGBoost classifier for testing
Accuracy = evaluate_ xgboost (xg_classifier, test_set)
# Returns the accuracy and XGBoost classifier
Return xg_classifier,accuracy
```

4. Pseudo code of KNN

```
function KNN (covid-19_dataset):
#Split the dataset into training and testing sets
train_set, test_set = split_dataset(covid-19_datset)
# Extract the features from the training set
features = extract_feature(train_set, features)
#Train the KNN classifier
knn_classifer =  train_knn(train_set,features)
# Evaluate the KNN classifier for testing
Accuracy = evaluate_ knn (knn_classifier, test_set)
# Returns the accuracy and KNN classifier
Return knn_classifier,accuracy
```

5. Pseudo code of Gradient Boosting

```
function Gradient_Boosting(covid-19_dataset):
#Split the dataset into training and testing sets
train_set, test_set = split_dataset(covid-19_datset)
# Extract the features for the training set
features = extract_feature(train_set, features)
#Train the Gradient Boosting classifier
gb_classifer =  train_gb(train_set,features)
# Evaluate the Gradient Boosting classifier for testing
Accuracy = evaluate_ gb (gb_classifier, test_set)
# Returns the accuracy and Gradient Boosting classifier
Return gb_classifier,accuracy
```

6. Pseudo code of Decision Tree

```
function Decision_Tree(covid-19_dataset):
#Split the dataset into training and testing sets
train_set, test_set = split_dataset(covid-19_datset)
# Extract the features for the training set
features = extract_feature(train_set, features)
#Train the Decision Tree classifier
dt_classifer = train_dt(train_set,features)
# Evaluate the Decision Tree classifier for testing
Accuracy = evaluate_ dt (dt_classifier, test_set)
# Returns the accuracy and Decision Tree classifier
Return dt_classifier,accuracy
```

7. Pseudo code of Neural Network

```
function Neural_Network(covid-19_dataset):
#Split the dataset into training and testing sets
train_set, test_set = split_dataset(covid-19_datset)
# Extract the features for the training set
features = extract_feature(train_set, features)
#Train the Neural Network classifier
nn_classifer = train_nn(train_set,features)
```

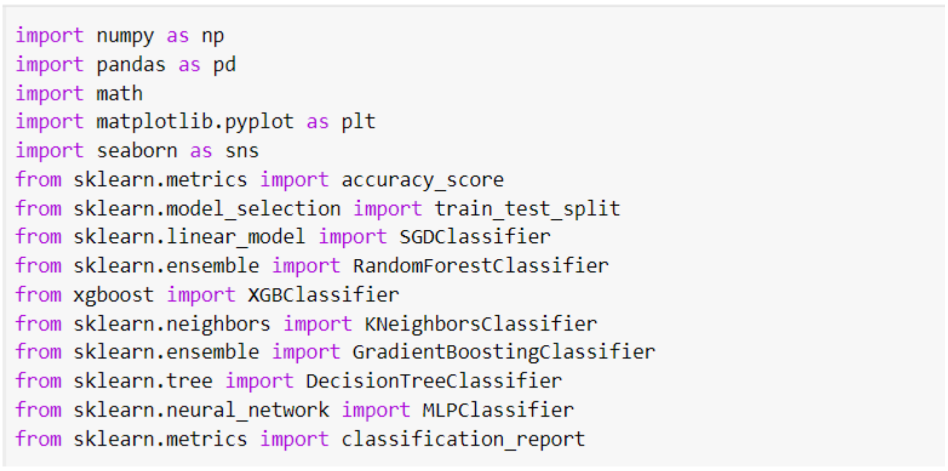
# Evaluate the Neural Network classifier for testing

Accuracy = evaluate_ nn (nn_classifier, test_set)

# Returns the accuracy and Neural Network classifier

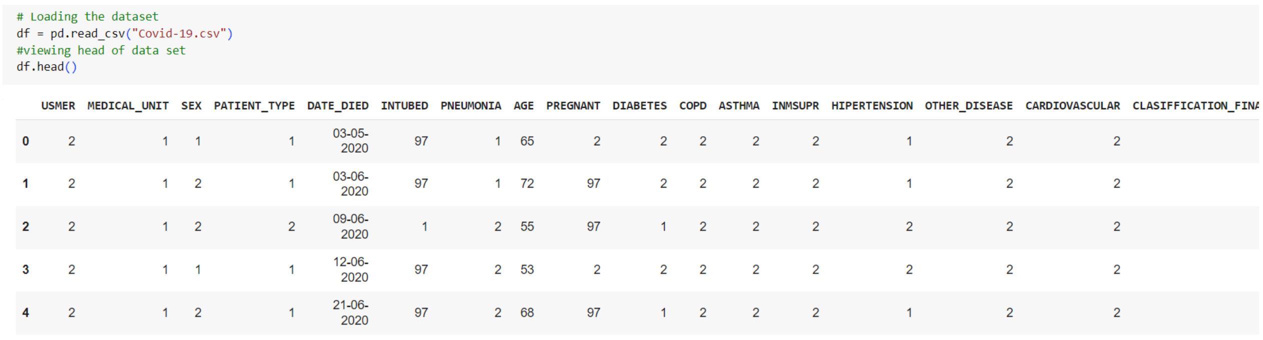
Return nn_classifier,accuracy

3.4.2 CODE SNIPPETS



```
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report
```

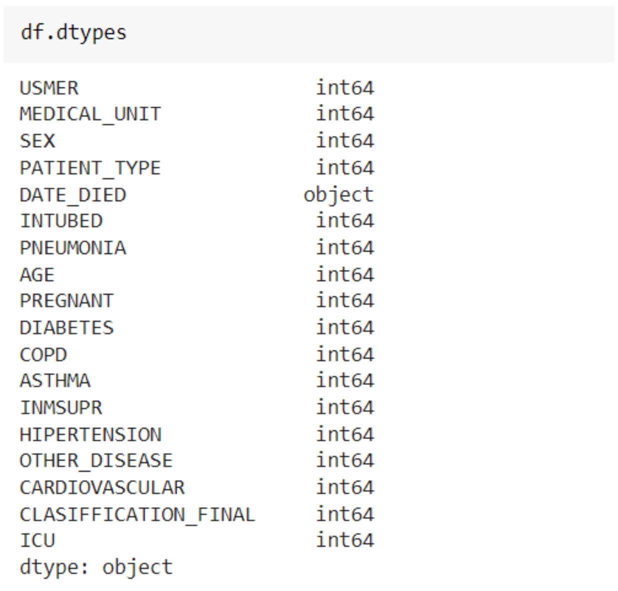
Fig 3.7 Importing Important libraries used



```
# Loading the dataset
df = pd.read_csv("Covid-19.csv")
#viewing head of data set
df.head()
```

	USMER	MEDICAL_UNIT	SEX	PATIENT_TYPE	DATE_DIED	INTUBED	PNEUMONIA	AGE	PREGNANT	DIABETES	COPD	ASTHMA	INMSUPR	HIPERTENSION	OTHER_DISEASE	CARDIOVASCULAR	CLASIFFICATION_FINA
0	2	1	1	1	03-05-2020	97	1	65	2	2	2	2	2	1	2	2	
1	2	1	2	1	03-06-2020	97	1	72	97	2	2	2	2	1	2	2	
2	2	1	2	2	09-06-2020	1	2	55	97	1	2	2	2	2	2	2	
3	2	1	1	1	12-06-2020	97	2	53	2	2	2	2	2	2	2	2	
4	2	1	2	1	21-06-2020	97	2	68	97	1	2	2	2	1	2	2	

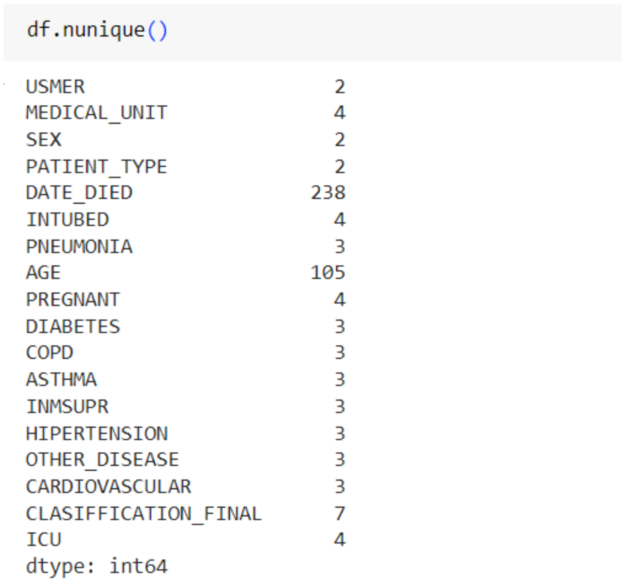
Fig 3.8 Loading of dataset



```
df.dtypes
```

```
USMER                   int64
MEDICAL_UNIT            int64
SEX                     int64
PATIENT_TYPE            int64
DATE_DIED              object
INTUBED                 int64
PNEUMONIA               int64
AGE                     int64
PREGNANT                int64
DIABETES                int64
COPD                    int64
ASTHMA                  int64
INMSUPR                 int64
HIPERTENSION            int64
OTHER_DISEASE           int64
CARDIOVASCULAR          int64
CLASIFFICATION_FINAL    int64
ICU                     int64
dtype: object
```

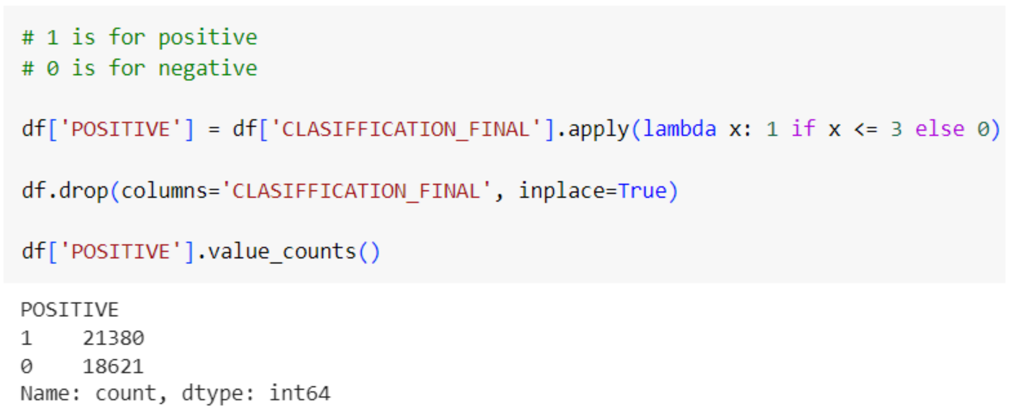
Fig 3.9 Showing data type of columns



```
df.nunique()
```

```
USMER                     2
MEDICAL_UNIT              4
SEX                       2
PATIENT_TYPE              2
DATE_DIED               238
INTUBED                   4
PNEUMONIA                 3
AGE                     105
PREGNANT                  4
DIABETES                  3
COPD                      3
ASTHMA                    3
INMSUPR                   3
HIPERTENSION              3
OTHER_DISEASE             3
CARDIOVASCULAR            3
CLASIFFICATION_FINAL      7
ICU                       4
dtype: int64
```

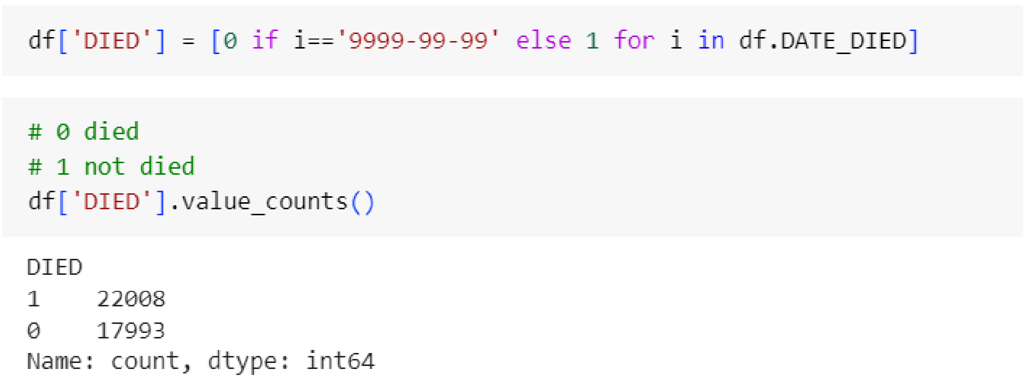
Fig 3.10 Getting the unique values in every column



```python
# 1 is for positive
# 0 is for negative

df['POSITIVE'] = df['CLASIFFICATION_FINAL'].apply(lambda x: 1 if x <= 3 else 0)

df.drop(columns='CLASIFFICATION_FINAL', inplace=True)

df['POSITIVE'].value_counts()
```

```
POSITIVE
1    21380
0    18621
Name: count, dtype: int64
```

Fig 3.11 Changing the name of column and converting the values into 0 and 1

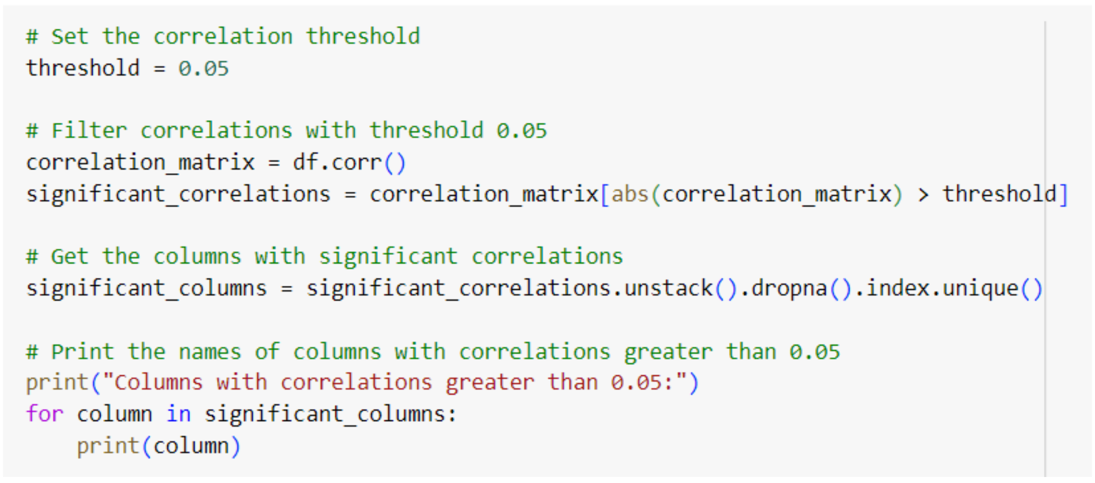


```python
df['DIED'] = [0 if i=='9999-99-99' else 1 for i in df.DATE_DIED]
```

```python
# 0 died
# 1 not died
df['DIED'].value_counts()
```

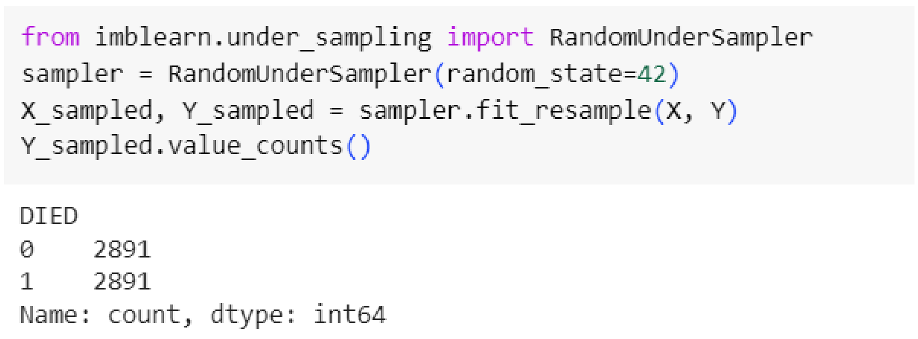
```
DIED
1    22008
0    17993
Name: count, dtype: int64
```

Fig 3.12 Count of 0 and 1 in died column



```python
# Set the correlation threshold
threshold = 0.05

# Filter correlations with threshold 0.05
correlation_matrix = df.corr()
significant_correlations = correlation_matrix[abs(correlation_matrix) > threshold]

# Get the columns with significant correlations
significant_columns = significant_correlations.unstack().dropna().index.unique()

# Print the names of columns with correlations greater than 0.05
print("Columns with correlations greater than 0.05:")
for column in significant_columns:
    print(column)
```

Fig 3.13 Feature selection with corelation coefficient



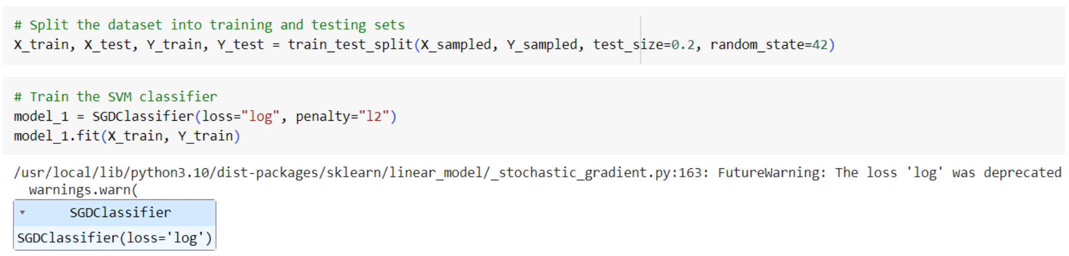
```
from imblearn.under_sampling import RandomUnderSampler
sampler = RandomUnderSampler(random_state=42)
X_sampled, Y_sampled = sampler.fit_resample(X, Y)
Y_sampled.value_counts()
```

```
DIED
0    2891
1    2891
Name: count, dtype: int64
```

Fig 3.14 Resampling of data for balancing the dataset

Models Used

a. Stochastic Gradient Descent

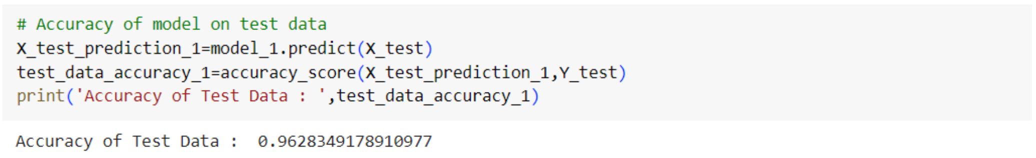


```
# Split the dataset into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X_sampled, Y_sampled, test_size=0.2, random_state=42)
```

```
# Train the SVM classifier
model_1 = SGDClassifier(loss="log", penalty="l2")
model_1.fit(X_train, Y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_stochastic_gradient.py:163: FutureWarning: The loss 'log' was deprecated
  warnings.warn(
▾     SGDClassifier
SGDClassifier(loss='log')
```

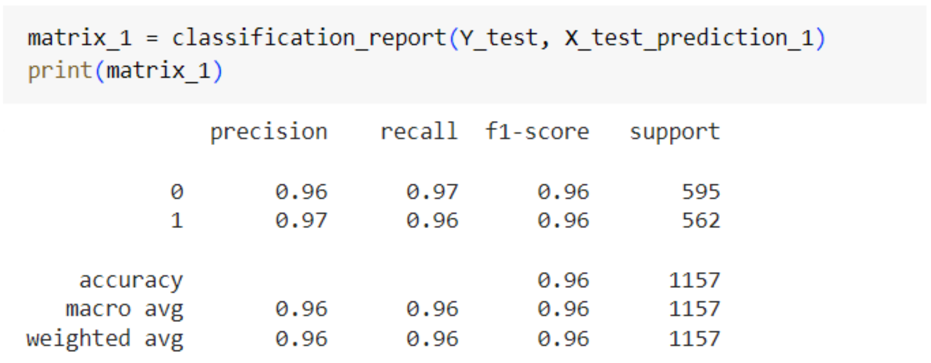
Fig 3.15 Training and fitting the SGD



```
# Accuracy of model on test data
X_test_prediction_1=model_1.predict(X_test)
test_data_accuracy_1=accuracy_score(X_test_prediction_1,Y_test)
print('Accuracy of Test Data : ',test_data_accuracy_1)
```

```
Accuracy of Test Data :  0.9628349178910977
```

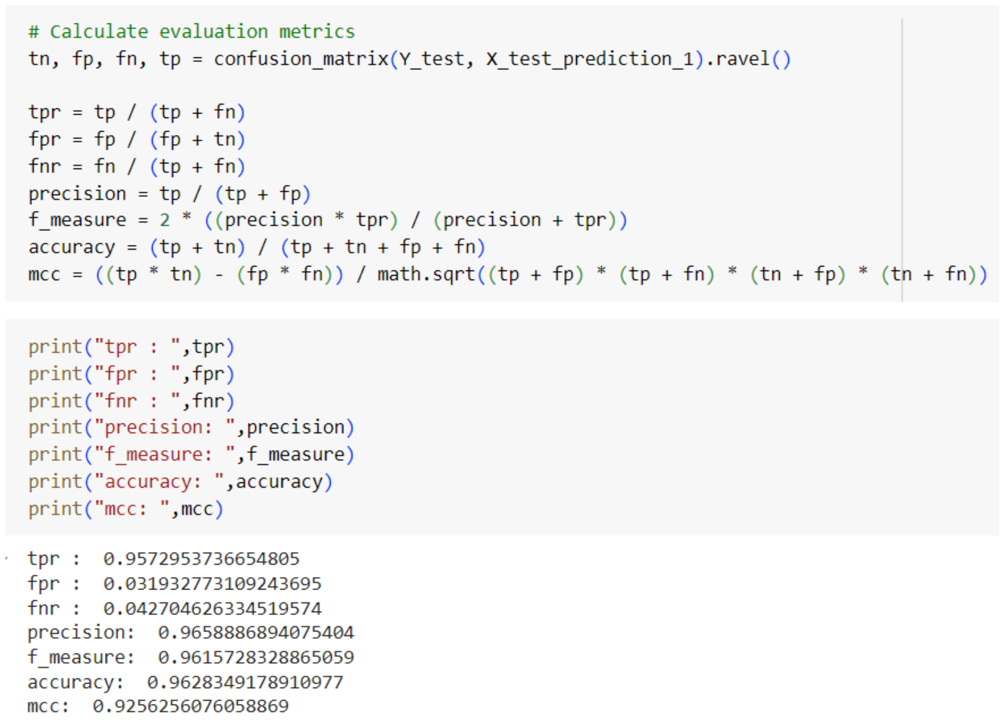
Fig 3.16 Computing the accuracy



```
matrix_1 = classification_report(Y_test, X_test_prediction_1)
print(matrix_1)
```

```
              precision    recall  f1-score   support

           0       0.96      0.97      0.96       595
           1       0.97      0.96      0.96       562

    accuracy                           0.96      1157
   macro avg       0.96      0.96      0.96      1157
weighted avg       0.96      0.96      0.96      1157
```

Fig 3.17 Classification report of the SGD used



```
# Calculate evaluation metrics
tn, fp, fn, tp = confusion_matrix(Y_test, X_test_prediction_1).ravel()

tpr = tp / (tp + fn)
fpr = fp / (fp + tn)
fnr = fn / (tp + fn)
precision = tp / (tp + fp)
f_measure = 2 * ((precision * tpr) / (precision + tpr))
accuracy = (tp + tn) / (tp + tn + fp + fn)
mcc = ((tp * tn) - (fp * fn)) / math.sqrt((tp + fp) * (tp + fn) * (tn + fp) * (tn + fn))
```

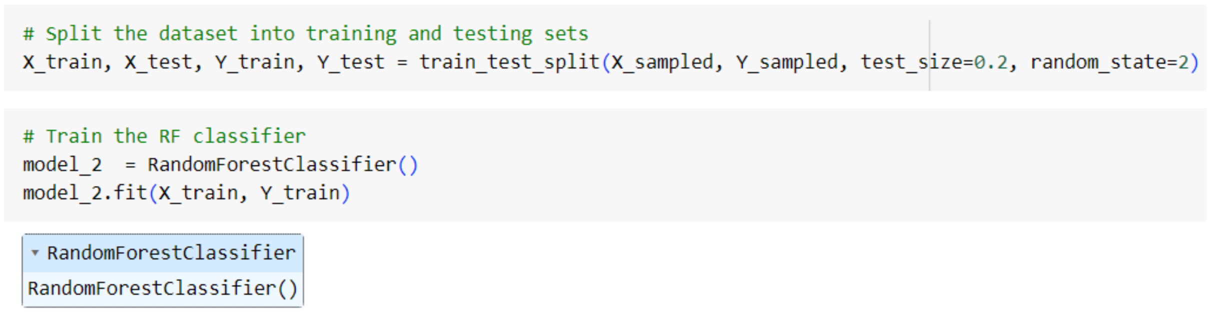
```
print("tpr : ",tpr)
print("fpr : ",fpr)
print("fnr : ",fnr)
print("precision: ",precision)
print("f_measure: ",f_measure)
print("accuracy: ",accuracy)
print("mcc: ",mcc)
```

```
tpr :  0.9572953736654805
fpr :  0.031932773109243695
fnr :  0.042704626334519574
precision:  0.9658886894075404
f_measure:  0.9615728328865059
accuracy:  0.9628349178910977
mcc:  0.9256256076058869
```

Fig 3.18 Overall calculation for the SGD

In figures 3.15 to 3.18 show the training part, fitting the model and the overall calculation regarding the SGD model used.

b. Random Forest

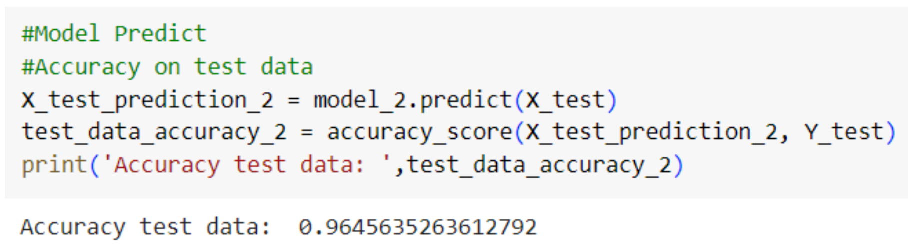


```
# Split the dataset into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X_sampled, Y_sampled, test_size=0.2, random_state=2)
```

```
# Train the RF classifier
model_2  = RandomForestClassifier()
model_2.fit(X_train, Y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

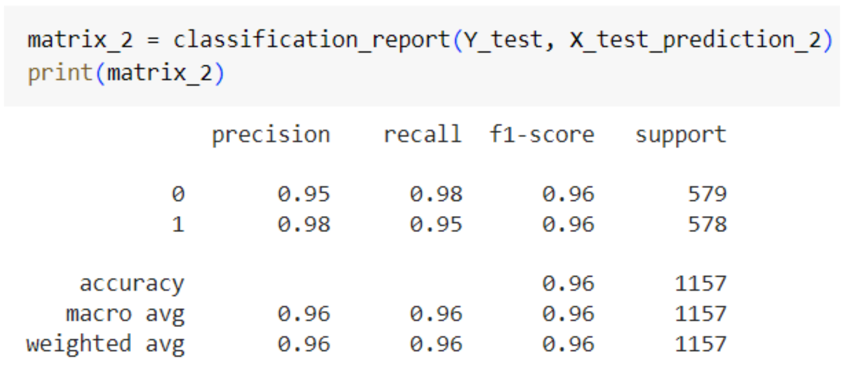
Fig 3.19 Training and fitting the RF



```
#Model Predict
#Accuracy on test data
X_test_prediction_2 = model_2.predict(X_test)
test_data_accuracy_2 = accuracy_score(X_test_prediction_2, Y_test)
print('Accuracy test data: ',test_data_accuracy_2)
```

```
Accuracy test data:  0.9645635263612792
```

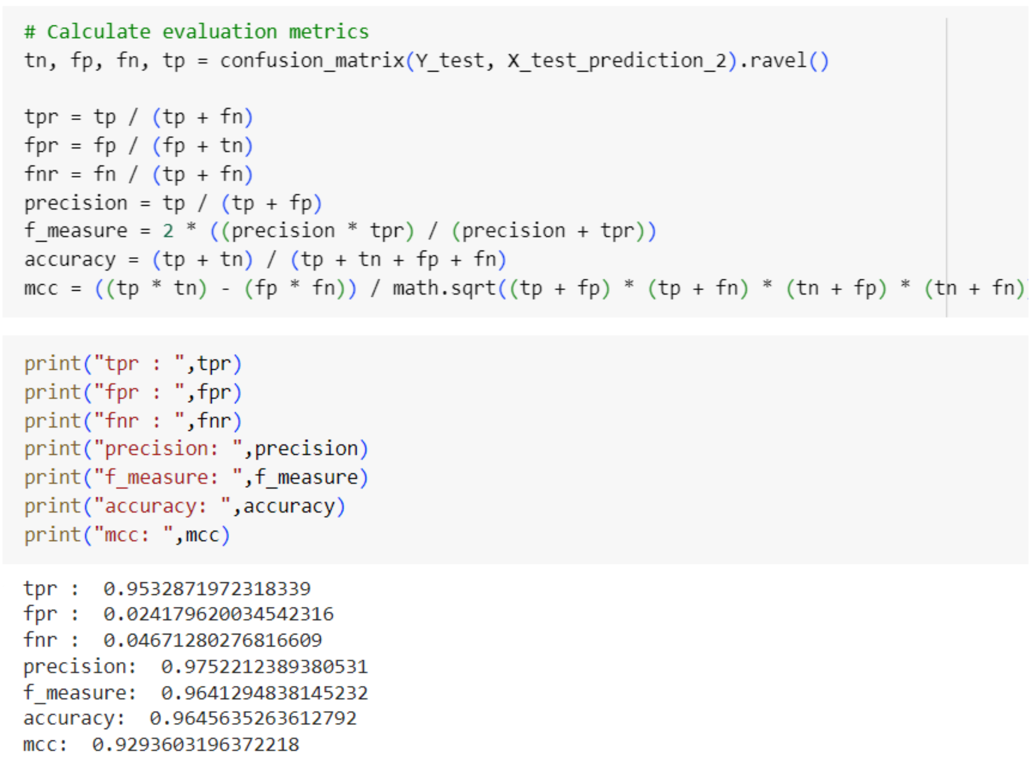
Fig 3.20 Computing the accuracy



```
matrix_2 = classification_report(Y_test, X_test_prediction_2)
print(matrix_2)
```

```
              precision    recall  f1-score   support

           0       0.95      0.98      0.96       579
           1       0.98      0.95      0.96       578

    accuracy                           0.96      1157
   macro avg       0.96      0.96      0.96      1157
weighted avg       0.96      0.96      0.96      1157
```

Fig 3.21 Classification report of the RF used



```
# Calculate evaluation metrics
tn, fp, fn, tp = confusion_matrix(Y_test, X_test_prediction_2).ravel()

tpr = tp / (tp + fn)
fpr = fp / (fp + tn)
fnr = fn / (tp + fn)
precision = tp / (tp + fp)
f_measure = 2 * ((precision * tpr) / (precision + tpr))
accuracy = (tp + tn) / (tp + tn + fp + fn)
mcc = ((tp * tn) - (fp * fn)) / math.sqrt((tp + fp) * (tp + fn) * (tn + fp) * (tn + fn)
```

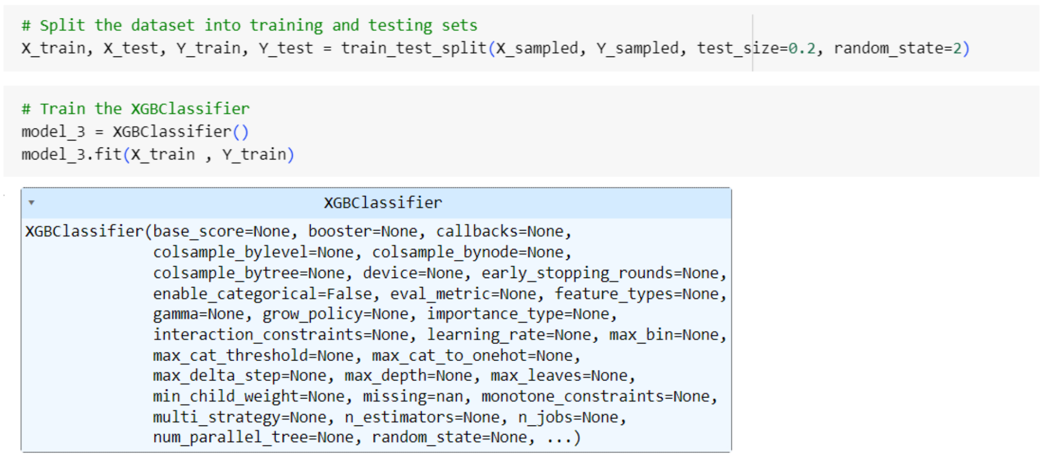
```
print("tpr : ",tpr)
print("fpr : ",fpr)
print("fnr : ",fnr)
print("precision: ",precision)
print("f_measure: ",f_measure)
print("accuracy: ",accuracy)
print("mcc: ",mcc)
```

```
tpr :  0.9532871972318339
fpr :  0.024179620034542316
fnr :  0.04671280276816609
precision:  0.9752212389380531
f_measure:  0.9641294838145232
accuracy:  0.9645635263612792
mcc:  0.9293603196372218
```

Fig 3.22 Overall calculation for the RF

In figures 3.19 to 3.22 show the training part, fitting the model and the overall calculation regarding the RF model used.

c. XGBoost

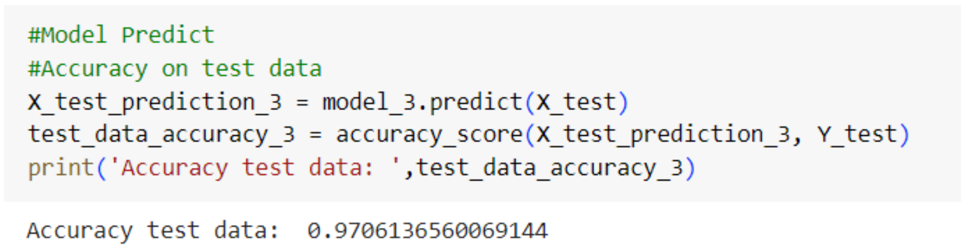


```
# Split the dataset into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X_sampled, Y_sampled, test_size=0.2, random_state=2)
```

```
# Train the XGBClassifier
model_3 = XGBClassifier()
model_3.fit(X_train , Y_train)
```

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

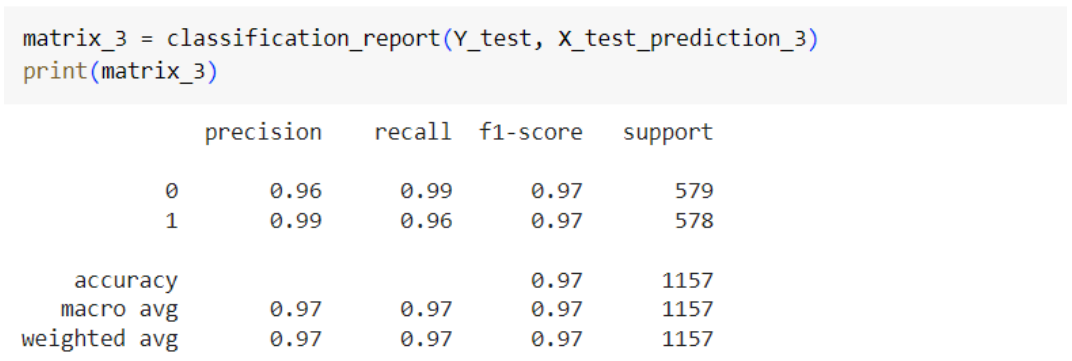
Fig 3.23 Training and fitting the XGBoost



```
#Model Predict
#Accuracy on test data
X_test_prediction_3 = model_3.predict(X_test)
test_data_accuracy_3 = accuracy_score(X_test_prediction_3, Y_test)
print('Accuracy test data: ',test_data_accuracy_3)
```

```
Accuracy test data:  0.9706136560069144
```

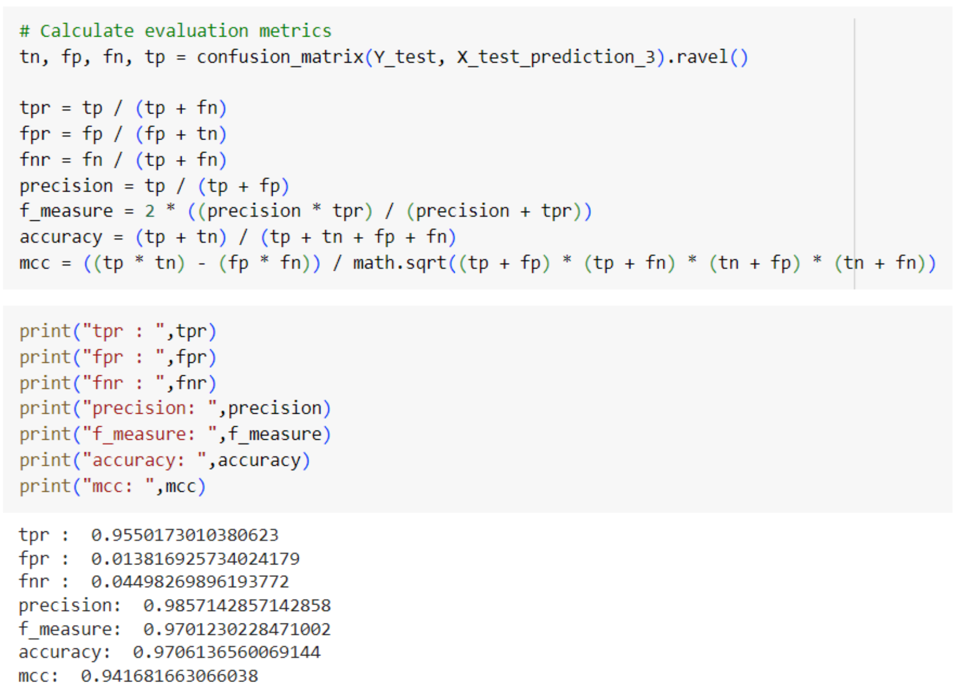
Fig 3.24 Computing the accuracy



```
matrix_3 = classification_report(Y_test, X_test_prediction_3)
print(matrix_3)
```

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.97       579
           1       0.99      0.96      0.97       578

    accuracy                           0.97      1157
   macro avg       0.97      0.97      0.97      1157
weighted avg       0.97      0.97      0.97      1157
```

Fig 3.25 Classification report of the XGBoost used



```
# Calculate evaluation metrics
tn, fp, fn, tp = confusion_matrix(Y_test, X_test_prediction_3).ravel()

tpr = tp / (tp + fn)
fpr = fp / (fp + tn)
fnr = fn / (tp + fn)
precision = tp / (tp + fp)
f_measure = 2 * ((precision * tpr) / (precision + tpr))
accuracy = (tp + tn) / (tp + tn + fp + fn)
mcc = ((tp * tn) - (fp * fn)) / math.sqrt((tp + fp) * (tp + fn) * (tn + fp) * (tn + fn))
```

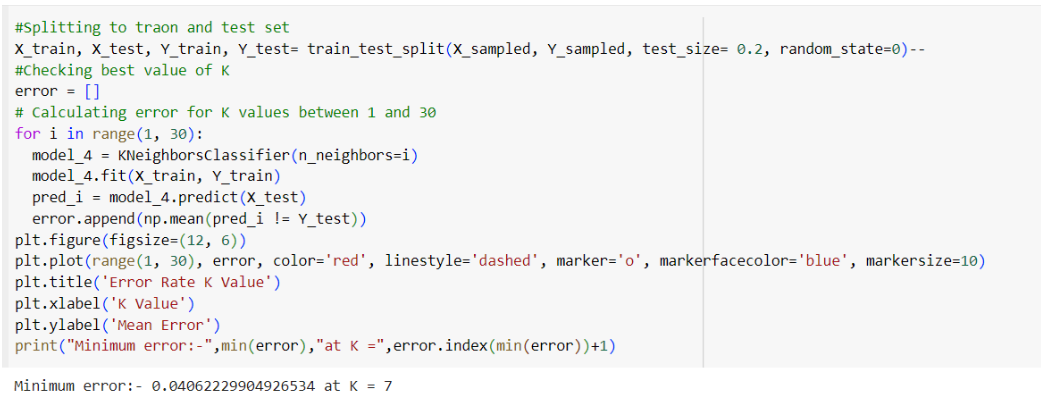
```
print("tpr : ",tpr)
print("fpr : ",fpr)
print("fnr : ",fnr)
print("precision: ",precision)
print("f_measure: ",f_measure)
print("accuracy: ",accuracy)
print("mcc: ",mcc)
```

```
tpr :  0.9550173010380623
fpr :  0.013816925734024179
fnr :  0.04498269896193772
precision:  0.9857142857142858
f_measure:  0.9701230228471002
accuracy:  0.9706136560069144
mcc:  0.941681663066038
```

Fig 3.26 Overall calculation for the XGBoost

In figures 3.23 to 3.26 show the training part, fitting the model and the overall calculation regarding the XGBoost model used.

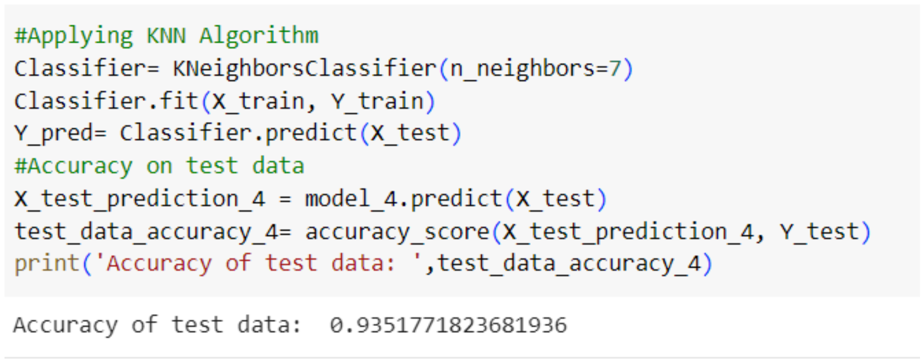
d. K-Nearst Neighbour



```
#Splitting to traon and test set
X_train, X_test, Y_train, Y_test= train_test_split(X_sampled, Y_sampled, test_size= 0.2, random_state=0)--
#Checking best value of K
error = []
# Calculating error for K values between 1 and 30
for i in range(1, 30):
  model_4 = KNeighborsClassifier(n_neighbors=i)
  model_4.fit(X_train, Y_train)
  pred_i = model_4.predict(X_test)
  error.append(np.mean(pred_i != Y_test))
plt.figure(figsize=(12, 6))
plt.plot(range(1, 30), error, color='red', linestyle='dashed', marker='o', markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
print("Minimum error:-",min(error),"at K =",error.index(min(error))+1)
```

```
Minimum error:- 0.04062229904926534 at K = 7
```

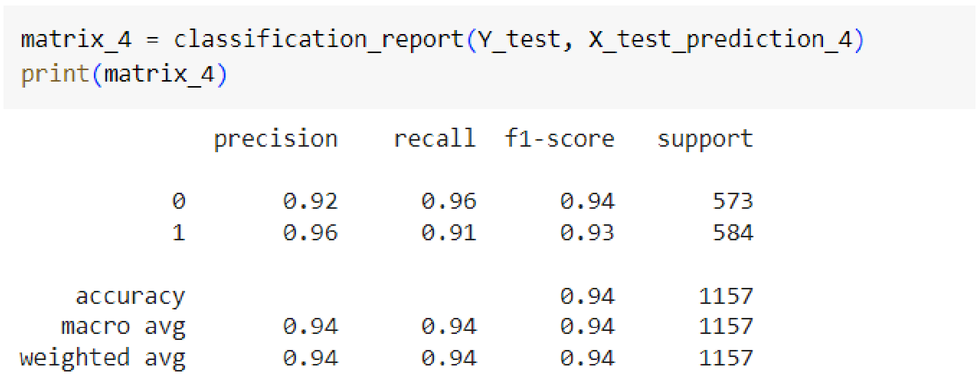
Fig 3.27 Training and fitting the model and getting the K value



```
#Applying KNN Algorithm
Classifier= KNeighborsClassifier(n_neighbors=7)
Classifier.fit(X_train, Y_train)
Y_pred= Classifier.predict(X_test)
#Accuracy on test data
X_test_prediction_4 = model_4.predict(X_test)
test_data_accuracy_4= accuracy_score(X_test_prediction_4, Y_test)
print('Accuracy of test data: ',test_data_accuracy_4)
```

```
Accuracy of test data:  0.9351771823681936
```

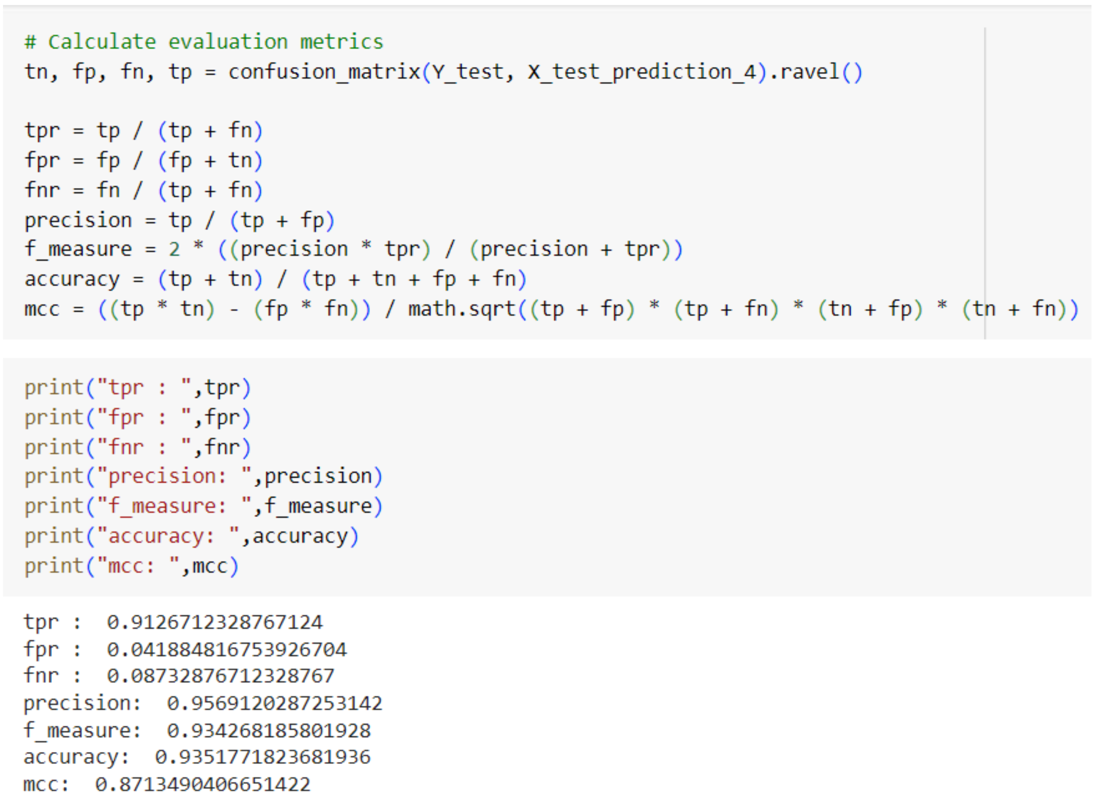
Fig 3.28 Computing the accuracy



```
matrix_4 = classification_report(Y_test, X_test_prediction_4)
print(matrix_4)
```

```
              precision    recall  f1-score   support

           0       0.92      0.96      0.94       573
           1       0.96      0.91      0.93       584

    accuracy                           0.94      1157
   macro avg       0.94      0.94      0.94      1157
weighted avg       0.94      0.94      0.94      1157
```

Fig 3.29 Classification report of the KNN used



```
# Calculate evaluation metrics
tn, fp, fn, tp = confusion_matrix(Y_test, X_test_prediction_4).ravel()

tpr = tp / (tp + fn)
fpr = fp / (fp + tn)
fnr = fn / (tp + fn)
precision = tp / (tp + fp)
f_measure = 2 * ((precision * tpr) / (precision + tpr))
accuracy = (tp + tn) / (tp + tn + fp + fn)
mcc = ((tp * tn) - (fp * fn)) / math.sqrt((tp + fp) * (tp + fn) * (tn + fp) * (tn + fn))
```

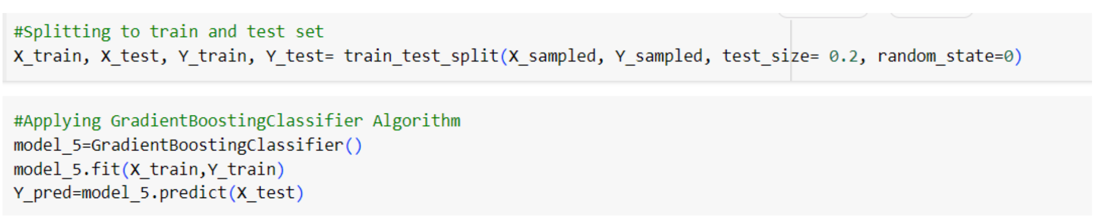
```
print("tpr : ",tpr)
print("fpr : ",fpr)
print("fnr : ",fnr)
print("precision: ",precision)
print("f_measure: ",f_measure)
print("accuracy: ",accuracy)
print("mcc: ",mcc)
```

```
tpr :  0.9126712328767124
fpr :  0.041884816753926704
fnr :  0.08732876712328767
precision:  0.9569120287253142
f_measure:  0.934268185801928
accuracy:  0.9351771823681936
mcc:  0.8713490406651422
```

Fig 3.30 Overall calculation for the KNN

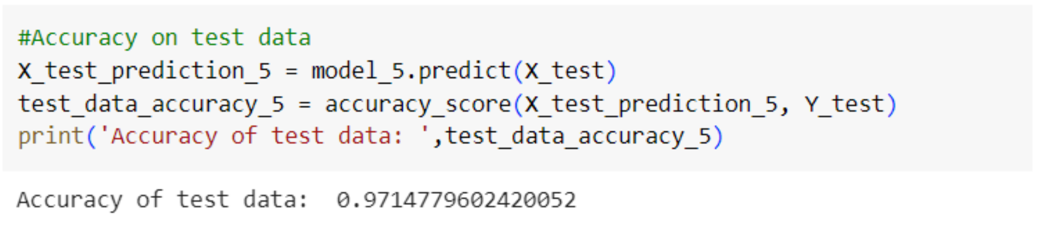
In figures 3.27 to 3.30 shows the training part and getting the value of k that is 7, fitting the model and the overall calculation regarding the KNN model used.

e. Gradient Boosting


```
#Splitting to train and test set
X_train, X_test, Y_train, Y_test= train_test_split(X_sampled, Y_sampled, test_size= 0.2, random_state=0)
```

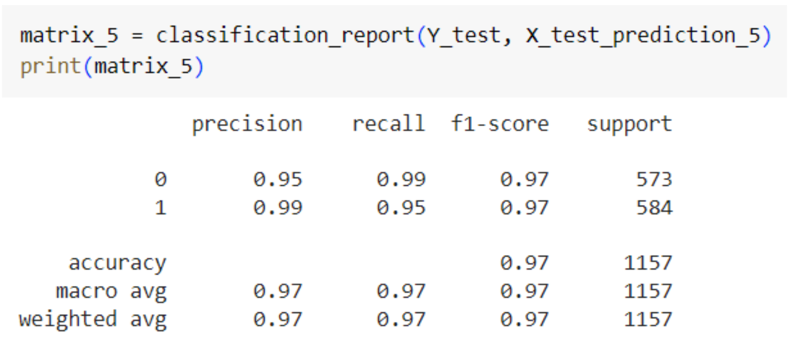
```
#Applying GradientBoostingClassifier Algorithm
model_5=GradientBoostingClassifier()
model_5.fit(X_train,Y_train)
Y_pred=model_5.predict(X_test)
```

Fig 3.31 Training and fitting the GB


```
#Accuracy on test data
X_test_prediction_5 = model_5.predict(X_test)
test_data_accuracy_5 = accuracy_score(X_test_prediction_5, Y_test)
print('Accuracy of test data: ',test_data_accuracy_5)
```

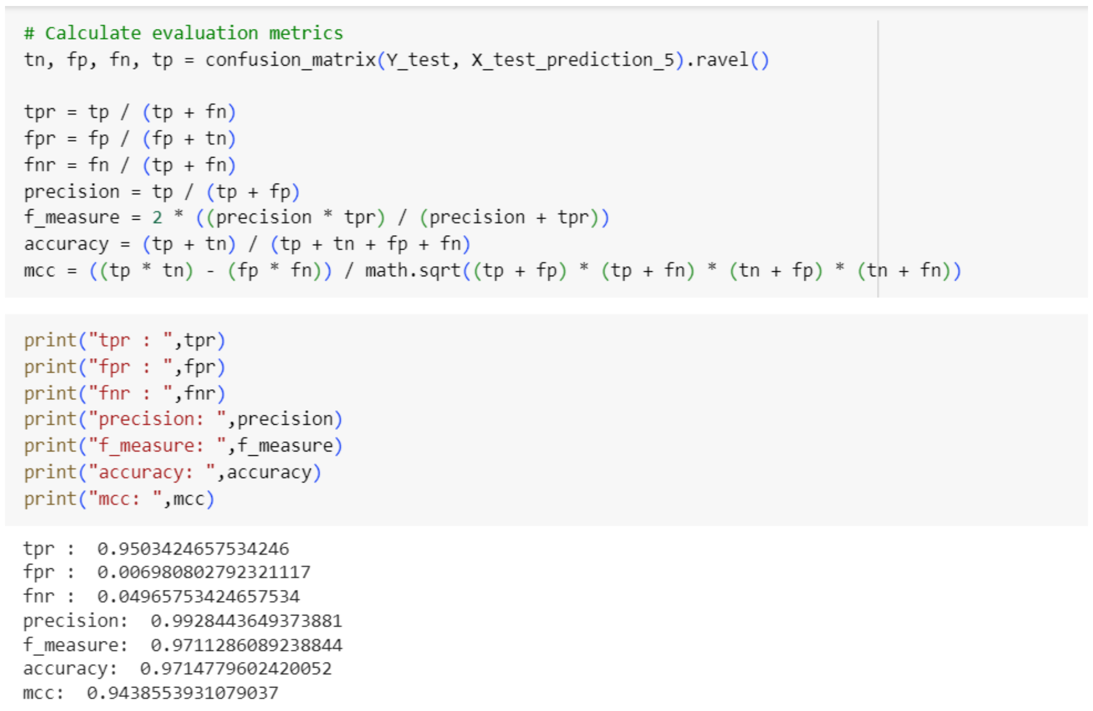
```
Accuracy of test data:  0.9714779602420052
```

Fig 3.32 Computing the accuracy


```
matrix_5 = classification_report(Y_test, X_test_prediction_5)
print(matrix_5)
```

```
              precision    recall  f1-score   support

           0       0.95      0.99      0.97       573
           1       0.99      0.95      0.97       584

    accuracy                           0.97      1157
   macro avg       0.97      0.97      0.97      1157
weighted avg       0.97      0.97      0.97      1157
```

Fig 3.33 Classification report of the GB used



```
# Calculate evaluation metrics
tn, fp, fn, tp = confusion_matrix(Y_test, X_test_prediction_5).ravel()

tpr = tp / (tp + fn)
fpr = fp / (fp + tn)
fnr = fn / (tp + fn)
precision = tp / (tp + fp)
f_measure = 2 * ((precision * tpr) / (precision + tpr))
accuracy = (tp + tn) / (tp + tn + fp + fn)
mcc = ((tp * tn) - (fp * fn)) / math.sqrt((tp + fp) * (tp + fn) * (tn + fp) * (tn + fn))
```

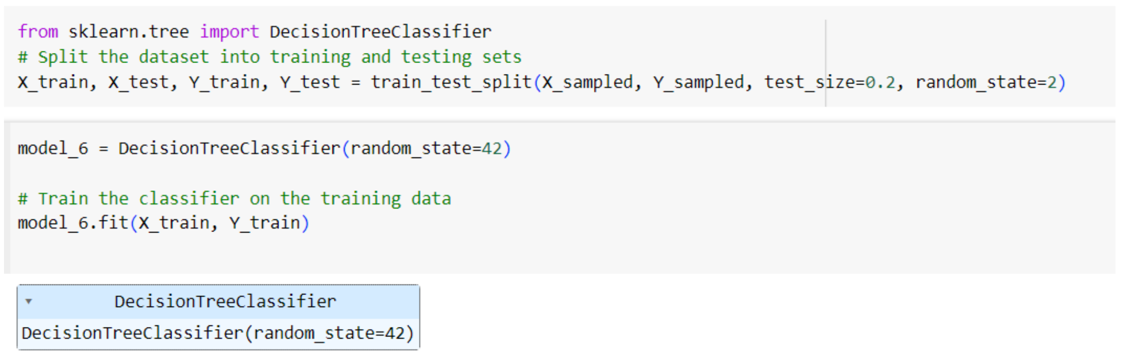
```
print("tpr : ",tpr)
print("fpr : ",fpr)
print("fnr : ",fnr)
print("precision: ",precision)
print("f_measure: ",f_measure)
print("accuracy: ",accuracy)
print("mcc: ",mcc)
```

```
tpr :  0.9503424657534246
fpr :  0.006980802792321117
fnr :  0.04965753424657534
precision:  0.9928443649373881
f_measure:  0.9711286089238844
accuracy:  0.9714779602420052
mcc:  0.9438553931079037
```

Fig 3.34 Overall calculation for the GB

In figures 3.31 to 3.34 shows the training part, fitting the model and the overall calculation regarding the GB model used.

f. Decision tree

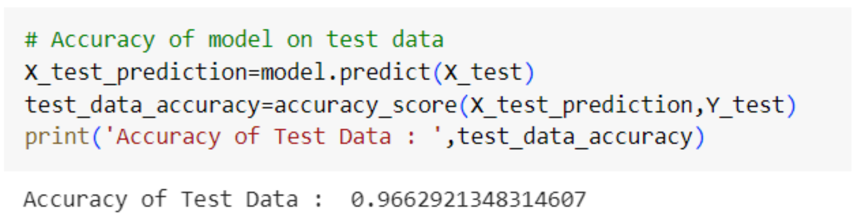


```
from sklearn.tree import DecisionTreeClassifier
# Split the dataset into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X_sampled, Y_sampled, test_size=0.2, random_state=2)
```

```
model_6 = DecisionTreeClassifier(random_state=42)

# Train the classifier on the training data
model_6.fit(X_train, Y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

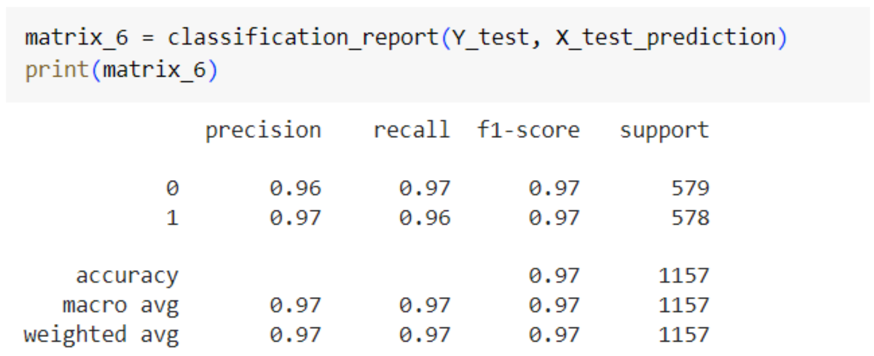
Fig 3.35 Training and fitting the DT



```
# Accuracy of model on test data
X_test_prediction=model.predict(X_test)
test_data_accuracy=accuracy_score(X_test_prediction,Y_test)
print('Accuracy of Test Data : ',test_data_accuracy)
```

```
Accuracy of Test Data :  0.9662921348314607
```

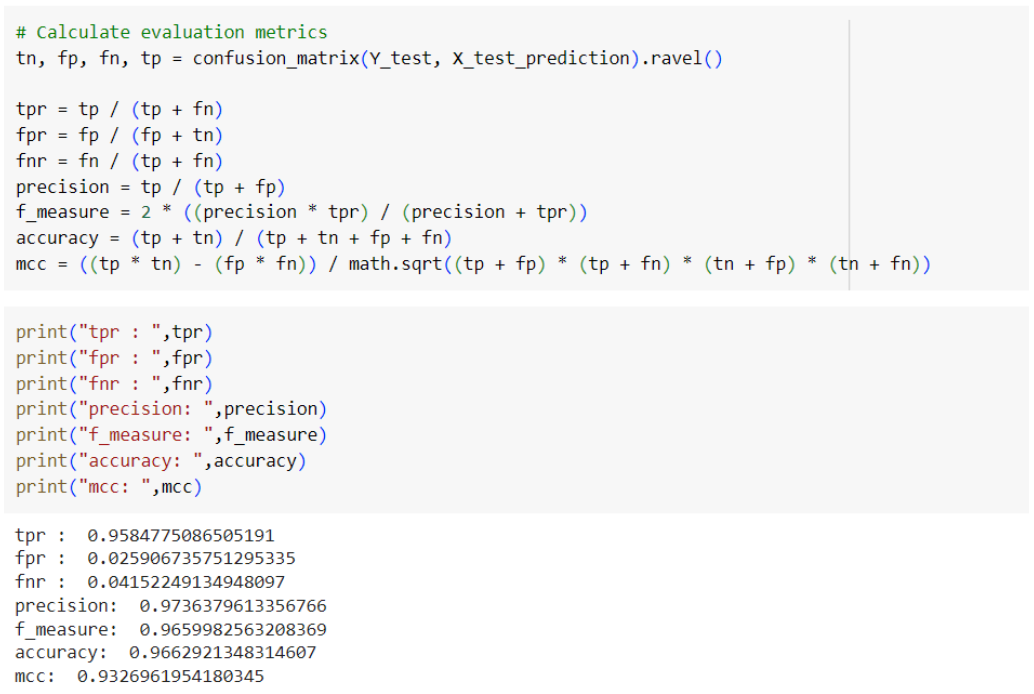
Fig 3.36 Computing the accuracy of DT



```
matrix_6 = classification_report(Y_test, X_test_prediction)
print(matrix_6)
```

```
              precision    recall  f1-score   support

           0       0.96      0.97      0.97       579
           1       0.97      0.96      0.97       578

    accuracy                           0.97      1157
   macro avg       0.97      0.97      0.97      1157
weighted avg       0.97      0.97      0.97      1157
```

Fig 3.37 Classification report of the DT used



```
# Calculate evaluation metrics
tn, fp, fn, tp = confusion_matrix(Y_test, X_test_prediction).ravel()

tpr = tp / (tp + fn)
fpr = fp / (fp + tn)
fnr = fn / (tp + fn)
precision = tp / (tp + fp)
f_measure = 2 * ((precision * tpr) / (precision + tpr))
accuracy = (tp + tn) / (tp + tn + fp + fn)
mcc = ((tp * tn) - (fp * fn)) / math.sqrt((tp + fp) * (tp + fn) * (tn + fp) * (tn + fn))
```

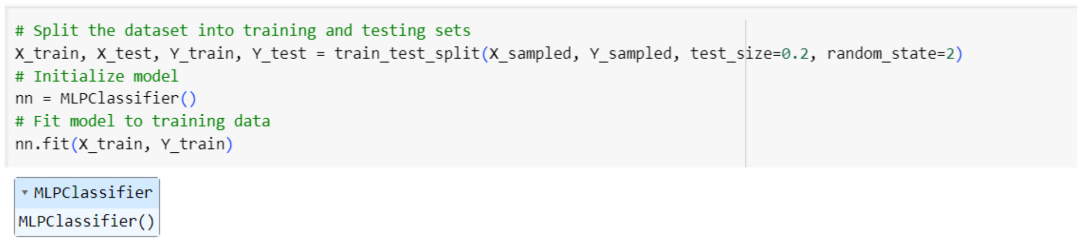
```
print("tpr : ",tpr)
print("fpr : ",fpr)
print("fnr : ",fnr)
print("precision: ",precision)
print("f_measure: ",f_measure)
print("accuracy: ",accuracy)
print("mcc: ",mcc)
```

```
tpr :  0.9584775086505191
fpr :  0.025906735751295335
fnr :  0.04152249134948097
precision:  0.9736379613356766
f_measure:  0.9659982563208369
accuracy:  0.9662921348314607
mcc:  0.9326961954180345
```

Fig 3.38 Overall calculation for the DT

In figures 3.35 to 3.38 shows the training part, fitting the model and the overall calculation regarding the DT model used.

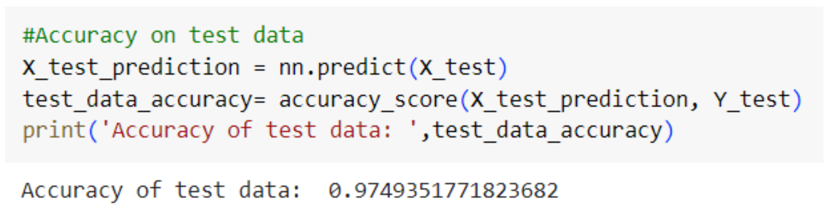
g. Neural Network



```
# Split the dataset into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X_sampled, Y_sampled, test_size=0.2, random_state=2)
# Initialize model
nn = MLPClassifier()
# Fit model to training data
nn.fit(X_train, Y_train)
```

```
▾ MLPClassifier
MLPClassifier()
```

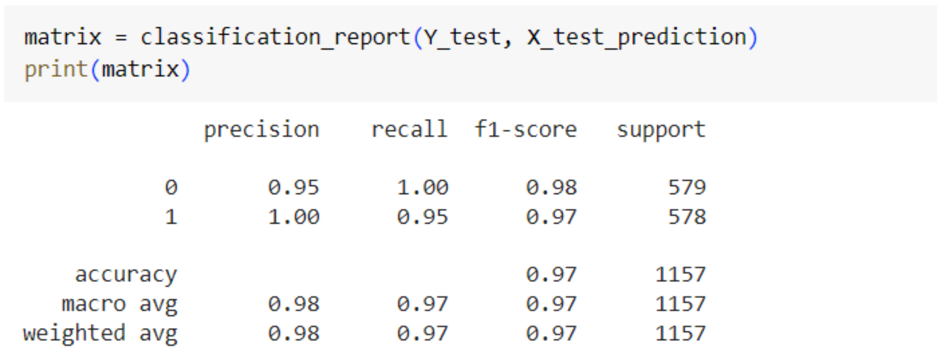
Fig 3.39 Training and fitting the NN



```
#Accuracy on test data
X_test_prediction = nn.predict(X_test)
test_data_accuracy= accuracy_score(X_test_prediction, Y_test)
print('Accuracy of test data: ',test_data_accuracy)
```

```
Accuracy of test data:  0.9749351771823682
```

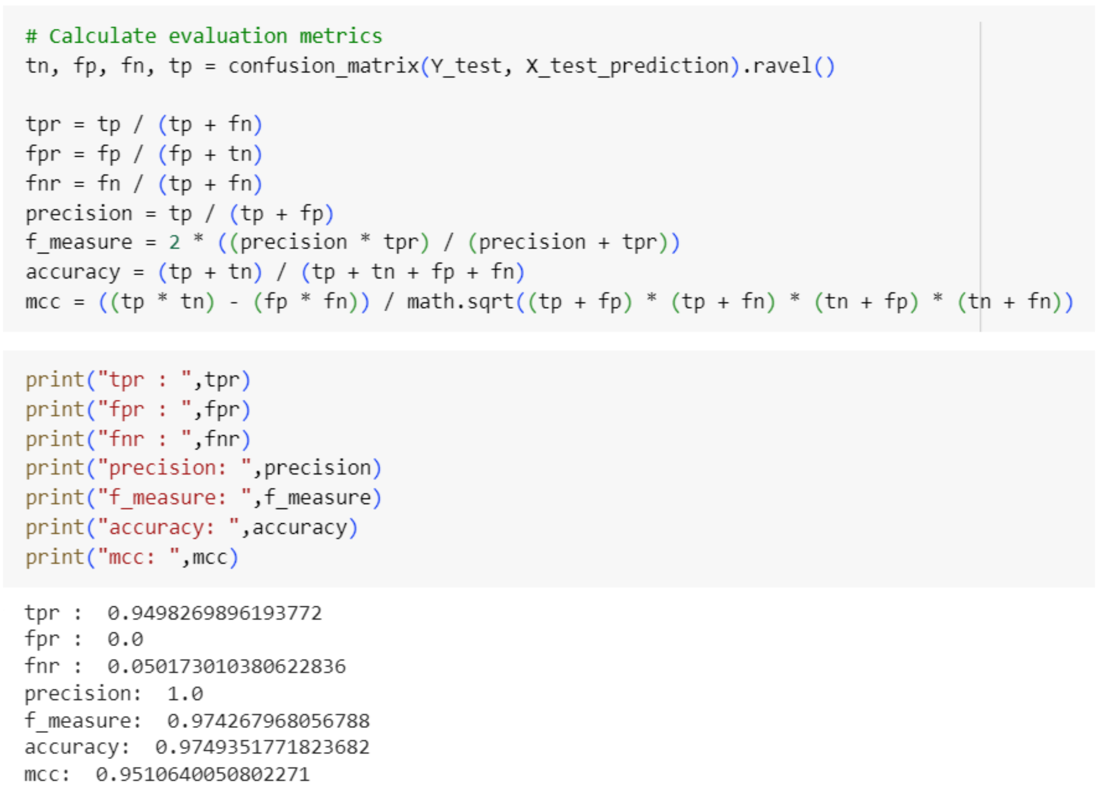
Fig 3.40 Computing the accuracy of NN



```
matrix = classification_report(Y_test, X_test_prediction)
print(matrix)
```

```
              precision    recall  f1-score   support

           0       0.95      1.00      0.98       579
           1       1.00      0.95      0.97       578

    accuracy                           0.97      1157
   macro avg       0.98      0.97      0.97      1157
weighted avg       0.98      0.97      0.97      1157
```

Fig 3.41 Classification report of the NN used



```
# Calculate evaluation metrics
tn, fp, fn, tp = confusion_matrix(Y_test, X_test_prediction).ravel()

tpr = tp / (tp + fn)
fpr = fp / (fp + tn)
fnr = fn / (tp + fn)
precision = tp / (tp + fp)
f_measure = 2 * ((precision * tpr) / (precision + tpr))
accuracy = (tp + tn) / (tp + tn + fp + fn)
mcc = ((tp * tn) - (fp * fn)) / math.sqrt((tp + fp) * (tp + fn) * (tn + fp) * (tn + fn))
```

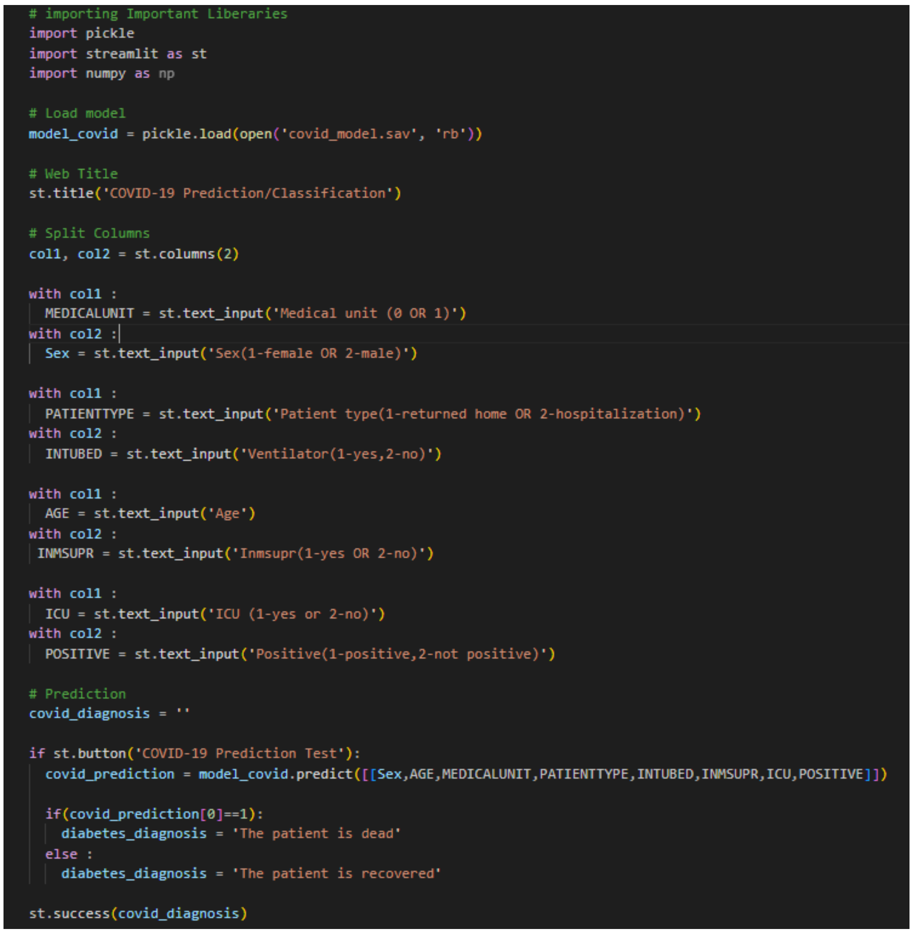
```
print("tpr : ",tpr)
print("fpr : ",fpr)
print("fnr : ",fnr)
print("precision: ",precision)
print("f_measure: ",f_measure)
print("accuracy: ",accuracy)
print("mcc: ",mcc)
```

```
tpr :  0.9498269896193772
fpr :  0.0
fnr :  0.050173010380622836
precision:  1.0
f_measure:  0.974267968056788
accuracy:  0.9749351771823682
mcc:  0.9510640050802271
```

Fig 3.42 Overall calculation for the NN

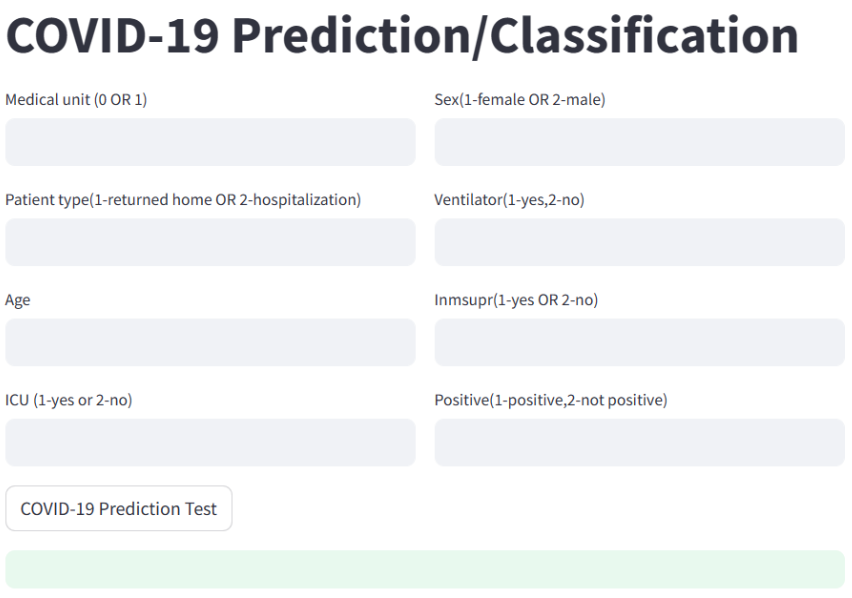
In figures 3.39 to 3.42 shows the training part, fitting the model and the overall calculation regarding the NN model used.

Web app



```python
# importing Important Liberaries
import pickle
import streamlit as st
import numpy as np

# Load model
model_covid = pickle.load(open('covid_model.sav', 'rb'))

# Web Title
st.title('COVID-19 Prediction/Classification')

# Split Columns
col1, col2 = st.columns(2)

with col1 :
  MEDICALUNIT = st.text_input('Medical unit (0 OR 1)')
with col2 :
  Sex = st.text_input('Sex(1-female OR 2-male)')

with col1 :
  PATIENTTYPE = st.text_input('Patient type(1-returned home OR 2-hospitalization)')
with col2 :
  INTUBED = st.text_input('Ventilator(1-yes,2-no)')

with col1 :
  AGE = st.text_input('Age')
with col2 :
 INMSUPR = st.text_input('Inmsupr(1-yes OR 2-no)')

with col1 :
  ICU = st.text_input('ICU (1-yes or 2-no)')
with col2 :
  POSITIVE = st.text_input('Positive(1-positive,2-not positive)')

# Prediction
covid_diagnosis = ''

if st.button('COVID-19 Prediction Test'):
  covid_prediction = model_covid.predict([[Sex,AGE,MEDICALUNIT,PATIENTTYPE,INTUBED,INMSUPR,ICU,POSITIVE]])

  if(covid_prediction[0]==1):
    diabetes_diagnosis = 'The patient is dead'
  else :
    diabetes_diagnosis = 'The patient is recovered'

st.success(covid_diagnosis)
```

Fig 3.43 Code for the Web app



COVID-19 Prediction/Classification

Medical unit (0 OR 1)

Sex(1-female OR 2-male)

Patient type(1-returned home OR 2-hospitalization)

Ventilator(1-yes,2-no)

Age

Inmsupr(1-yes OR 2-no)

ICU (1-yes or 2-no)

Positive(1-positive,2-not positive)

COVID-19 Prediction Test

Fig 3.44 Snippet of Web app

The Figures 3.43 describes the code that is used for the we app using streamlit and figure 3.44 shows the screenshot of the web app that is made using streamlit.

3.5 KEY CHALLENGES

There are different key challenges faced during the project

- Imbalanced dataset – The number of death and recovered cases are imbalanced and this can affect the model's ability to generalize. Also, it would learn toward predicting the majority class and struggle with the minority group.
- Data quality and Accessibility - Getting the accurate and reliable data for classification of covid-19 cases can be a challenging task as less data is available and inconsistency in the dataset.
- Feature Selection – Finding the important features and getting the meaningful variables for input can be complex and difficult. Also, deciding between the feature which contributes most for getting the outcome requires experimentation.
- Handling missing data – Dealing with the missing values in the data is crucial part. To ensure the models accurate working robust imputation techniques must be applied.
- Model Selection – Selecting the best models that performs best for the given data is complex and tedious task and also finding the patterns so that it can classify data accurately.
- Model Overfitting – It is a case in which model gives accuracy of the training data more and less test accuracy as it memorizes training data instead of learning different patterns associated.
- Relationship with other disease- The interaction between other infectious disease and finding the patterns can impact the modelling approaches.
- Educational and communicative hurdles – It is an ongoing challenge to bridge the gap in understanding the communication that is effective in predicting and classifying it to the diverse audience and healthcare department and other persons.

CHAPTER 4: TESTING

4.1 TESTING STRATEGY

Multiple inputs are given to the model to test its working as it is giving us the right output or not. To test the model and to make its performance good, we calculate the accuracy of the model and further we find other parameters such as f1-score, recall and precision to make the model best and efficient during its working [24]. To test the model, we use methods like test-train split, confusion-matrix, cross-validation and plotting different graphs to display the outcomes and the efficiency of the model those are described as

- Data splitting

  Training set – The major part of the data is used for training the machine learning models. This includes the historical data of symptoms of patients and other relevant features.

  Test Set – It is a completely independent set of data that used for the final model evaluation for the prediction. This part simulates real world scenarios where each model encounters new data that is not seen before.
- Performance metrics

  Accuracy – It is the percentage correctness of prediction that is correctly recognized death and recovered cases.

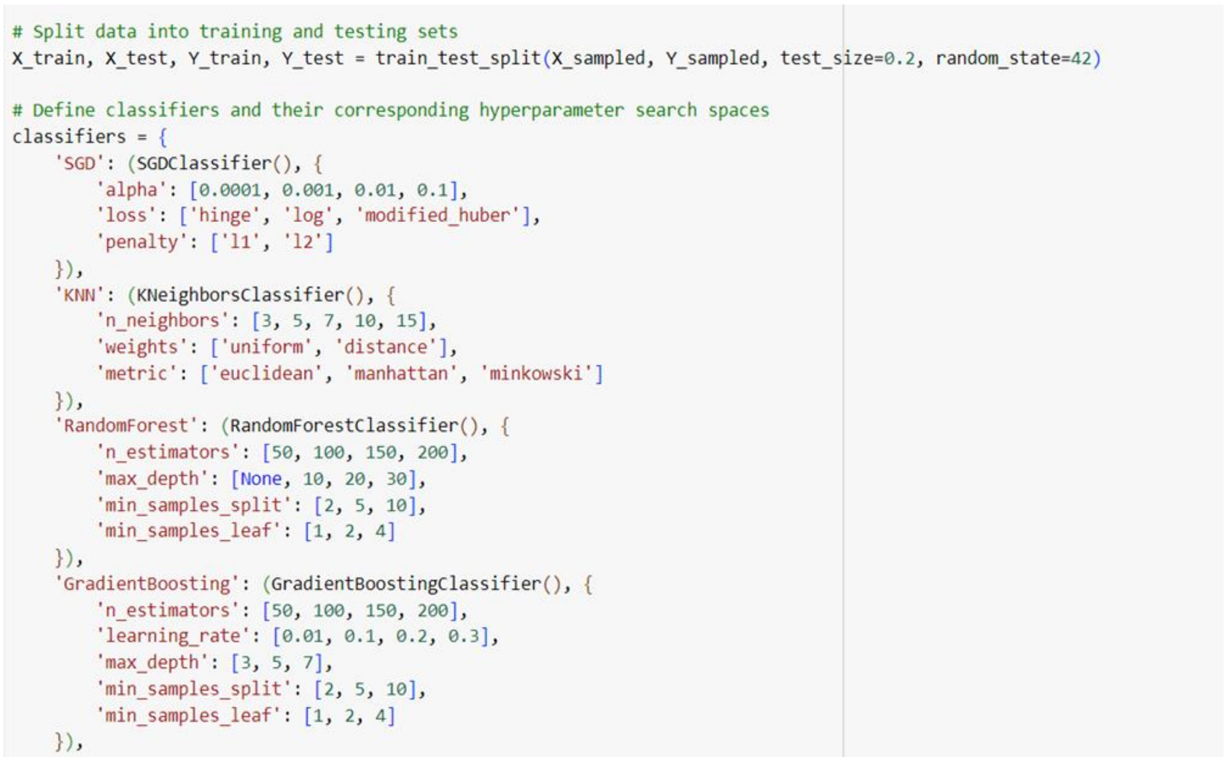
  Precision – It is the degree of exactness.

  F1 Score – It is the harmonic mean of precision and recall.

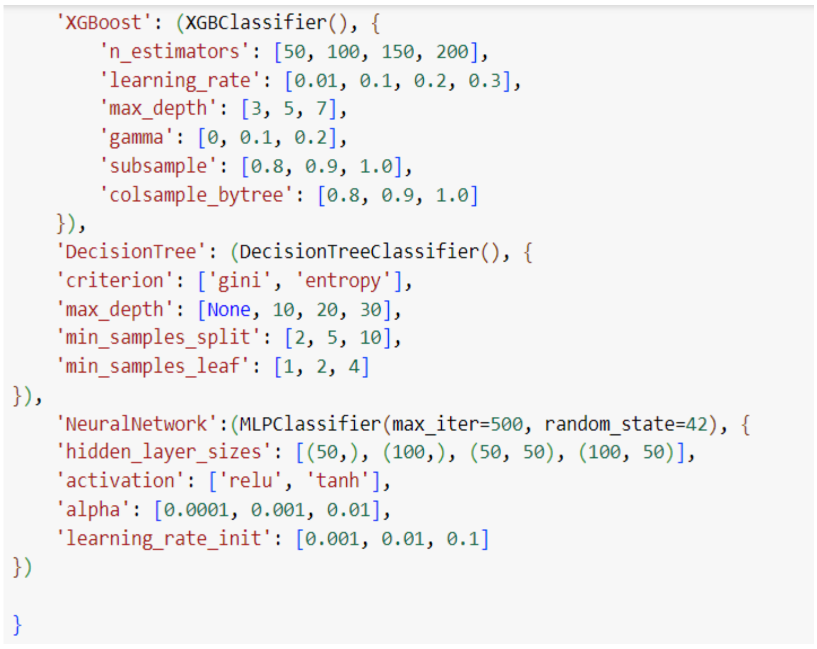
  AUC-ROC (Area Under the Receiver Operating Characteristic Curve) – Used for binary classification task to indicate the model's ability to differentiate between classes.
- Confusion matrix – It summarizes the model's performance by categorizing prediction into four parts such as True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).
- Hyperparameter tunning – It is used to increase the accuracy of the model by using certain best parameters.

4.2 TEST CASES AND OUTCOMES

The described outcomes show that we have applied hyperparameter tunning in which best values of different hyperparameter of different models are selected to increase their accuracy and other parameters.



```
# Split data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X_sampled, Y_sampled, test_size=0.2, random_state=42)

# Define classifiers and their corresponding hyperparameter search spaces
classifiers = {
    'SGD': (SGDClassifier(), {
        'alpha': [0.0001, 0.001, 0.01, 0.1],
        'loss': ['hinge', 'log', 'modified_huber'],
        'penalty': ['l1', 'l2']
    }),
    'KNN': (KNeighborsClassifier(), {
        'n_neighbors': [3, 5, 7, 10, 15],
        'weights': ['uniform', 'distance'],
        'metric': ['euclidean', 'manhattan', 'minkowski']
    }),
    'RandomForest': (RandomForestClassifier(), {
        'n_estimators': [50, 100, 150, 200],
        'max_depth': [None, 10, 20, 30],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4]
    }),
    'GradientBoosting': (GradientBoostingClassifier(), {
        'n_estimators': [50, 100, 150, 200],
        'learning_rate': [0.01, 0.1, 0.2, 0.3],
        'max_depth': [3, 5, 7],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4]
    }),
```



```
    'XGBoost': (XGBClassifier(), {
        'n_estimators': [50, 100, 150, 200],
        'learning_rate': [0.01, 0.1, 0.2, 0.3],
        'max_depth': [3, 5, 7],
        'gamma': [0, 0.1, 0.2],
        'subsample': [0.8, 0.9, 1.0],
        'colsample_bytree': [0.8, 0.9, 1.0]
    }),
    'DecisionTree': (DecisionTreeClassifier(), {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}),
    'NeuralNetwork':(MLPClassifier(max_iter=500, random_state=42), {
    'hidden_layer_sizes': [(50,), (100,), (50, 50), (100, 50)],
    'activation': ['relu', 'tanh'],
    'alpha': [0.0001, 0.001, 0.01],
    'learning_rate_init': [0.001, 0.01, 0.1]
})

}
```

Fig 4.1 Hyperparameter tunning of base classifiers

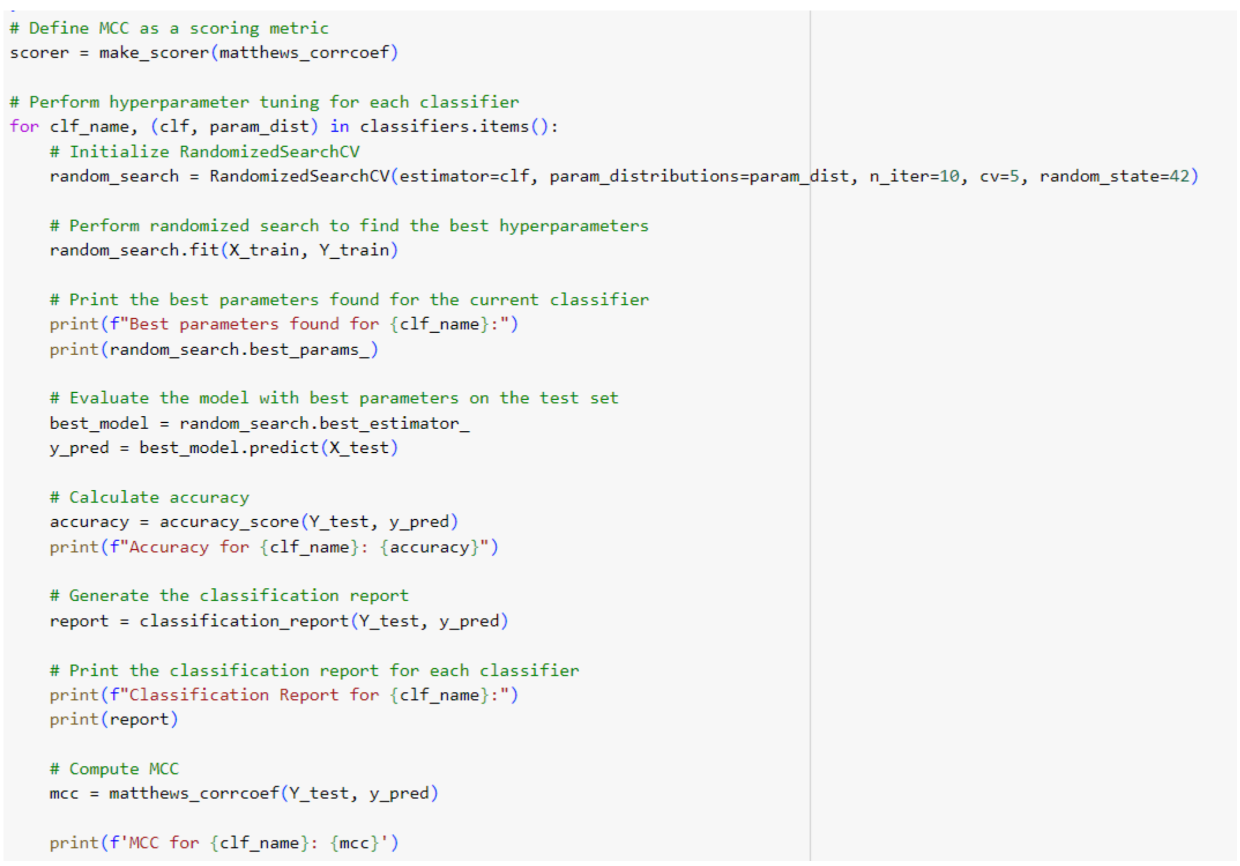

```
# Define MCC as a scoring metric
scorer = make_scorer(matthews_corrcoef)

# Perform hyperparameter tuning for each classifier
for clf_name, (clf, param_dist) in classifiers.items():
    # Initialize RandomizedSearchCV
    random_search = RandomizedSearchCV(estimator=clf, param_distributions=param_dist, n_iter=10, cv=5, random_state=42)

    # Perform randomized search to find the best hyperparameters
    random_search.fit(X_train, Y_train)

    # Print the best parameters found for the current classifier
    print(f"Best parameters found for {clf_name}:")
    print(random_search.best_params_)

    # Evaluate the model with best parameters on the test set
    best_model = random_search.best_estimator_
    y_pred = best_model.predict(X_test)

    # Calculate accuracy
    accuracy = accuracy_score(Y_test, y_pred)
    print(f"Accuracy for {clf_name}: {accuracy}")

    # Generate the classification report
    report = classification_report(Y_test, y_pred)

    # Print the classification report for each classifier
    print(f"Classification Report for {clf_name}:")
    print(report)

    # Compute MCC
    mcc = matthews_corrcoef(Y_test, y_pred)

    print(f'MCC for {clf_name}: {mcc}')
```

Fig 4.2 Describe the mcc and classification report of all the base classifiers

CHAPTER 5: RESULT AND EVALUATION

5.1 RESULT

We have analysed the results with several classifiers having different types of optimal parameters. The results demonstrates that Neural Network model plays a crucial role in the classification and prediction of covid – 19 death and recovered cases as it produces precise results compared to other models after that Gradient boosting also achieved the best accuracy as compared to other classifiers such as SGD, Random forest, KNN, XGBoost, and Decision tree. This Fig. 5.1 describes the age distribution that which part of the age is more affected those are between the age 40 to 75 approx. From Fig. 5.2 to 5.6 describe the data with respect to the features it has and how it can affect age with respect to covid classification, gender, other symptoms. Fig. 5.7 describes the features selection with respect to correlation. The Fig. 5.8 to 5.12 describes the confusion matrix with actual and predicted values for all the models applied.

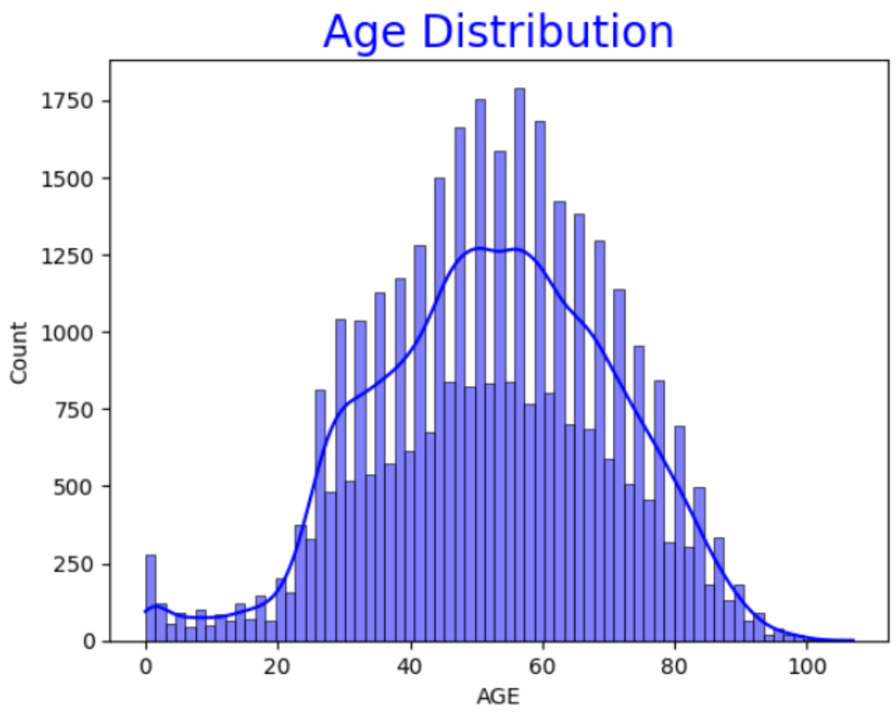


Fig 5.1 Display of age distribution

This Fig. 5.1 describes the age distribution that which part of the age is more affected those are between the age 40 to 75 approx.

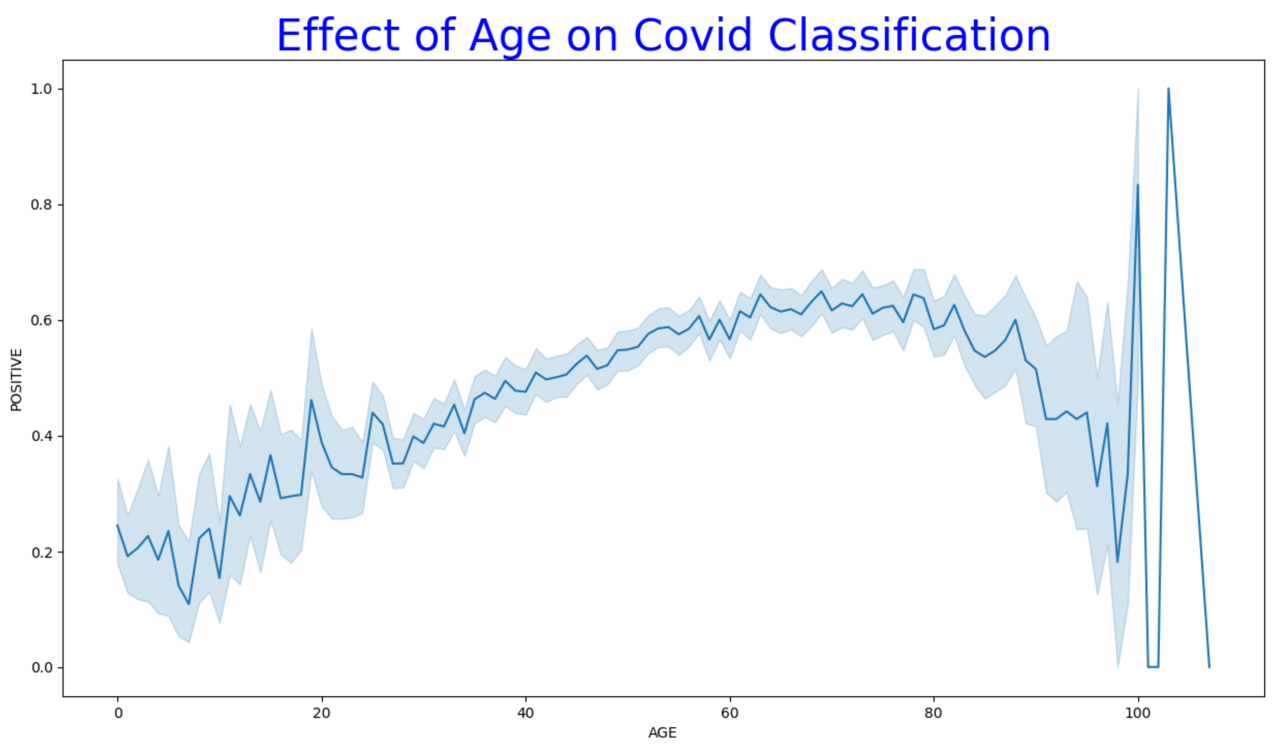


Fig. 5.2 Display effect of covid-19 on age

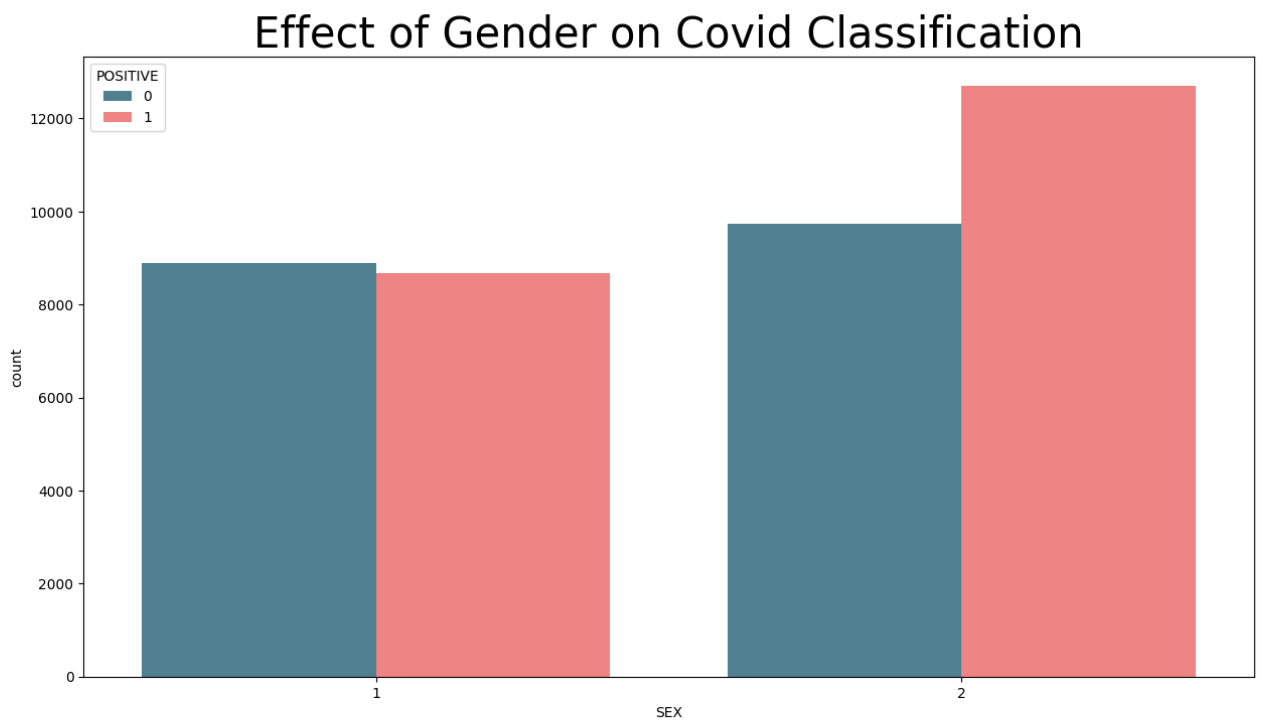


Fig 5.3 Shows the effect of covid-19 on gender.

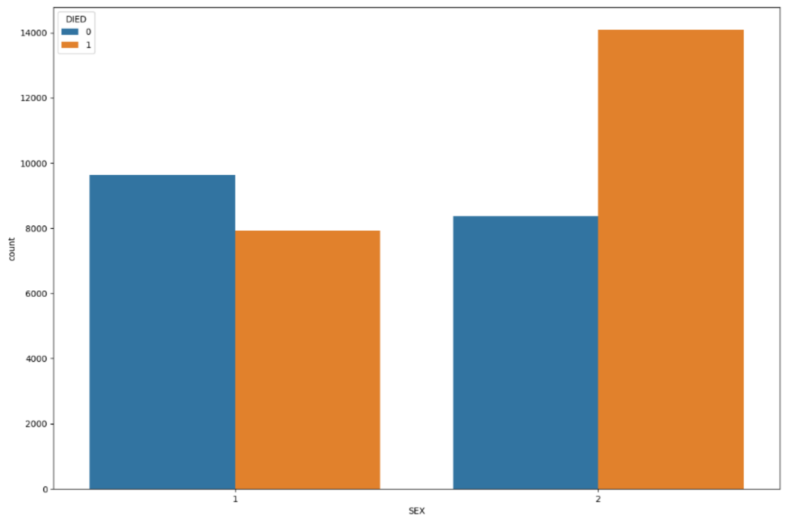


Fig. 5.4 Shows the effect number of positive cases.

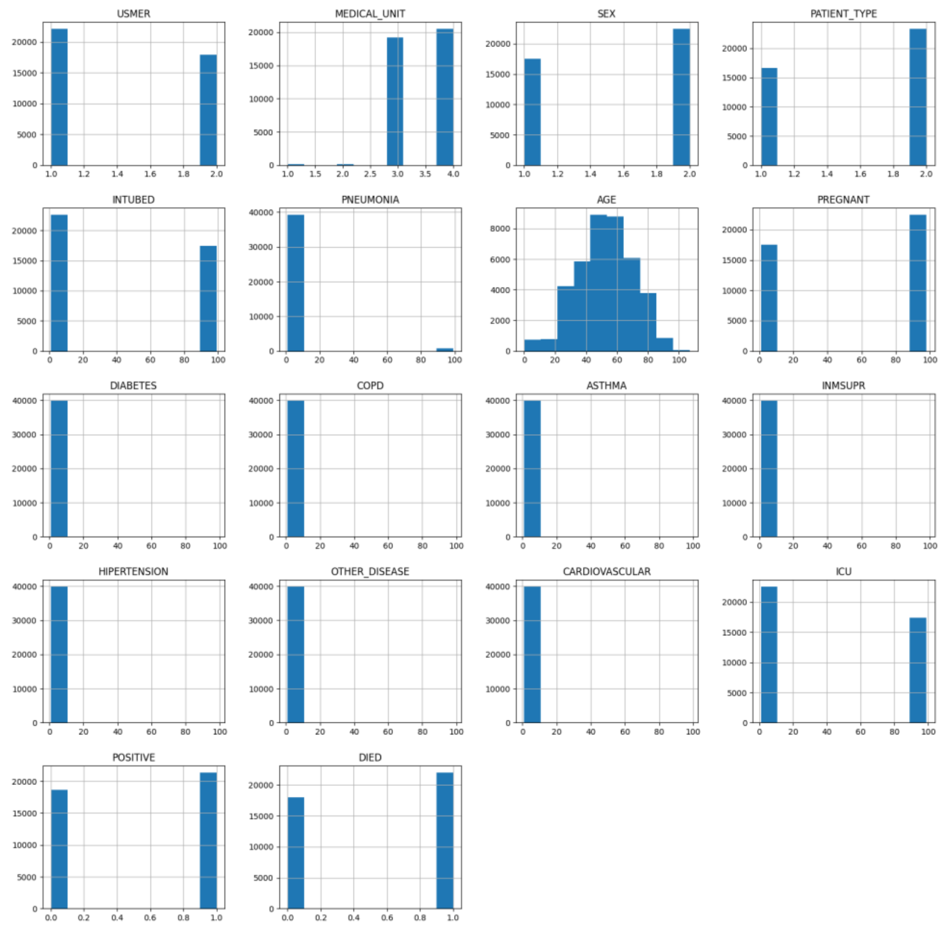


Fig. 5.5 Shows different features plot

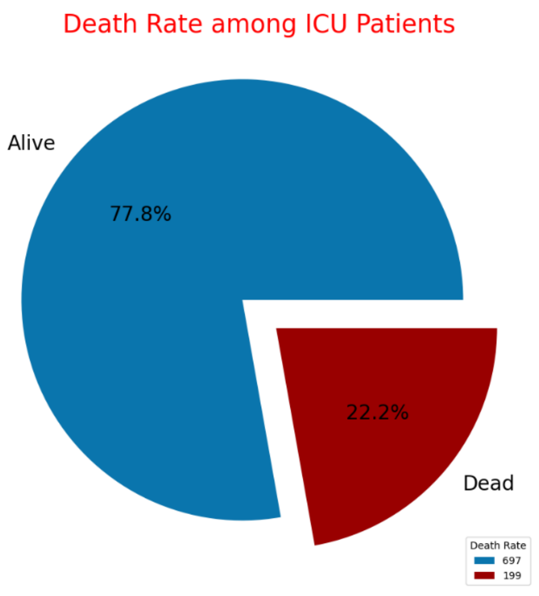


Fig. 5.6 Shows the death rate among the ICU patients.

Figure 5.2 to 5.6 visualize the features and comparison with other features using the graphs

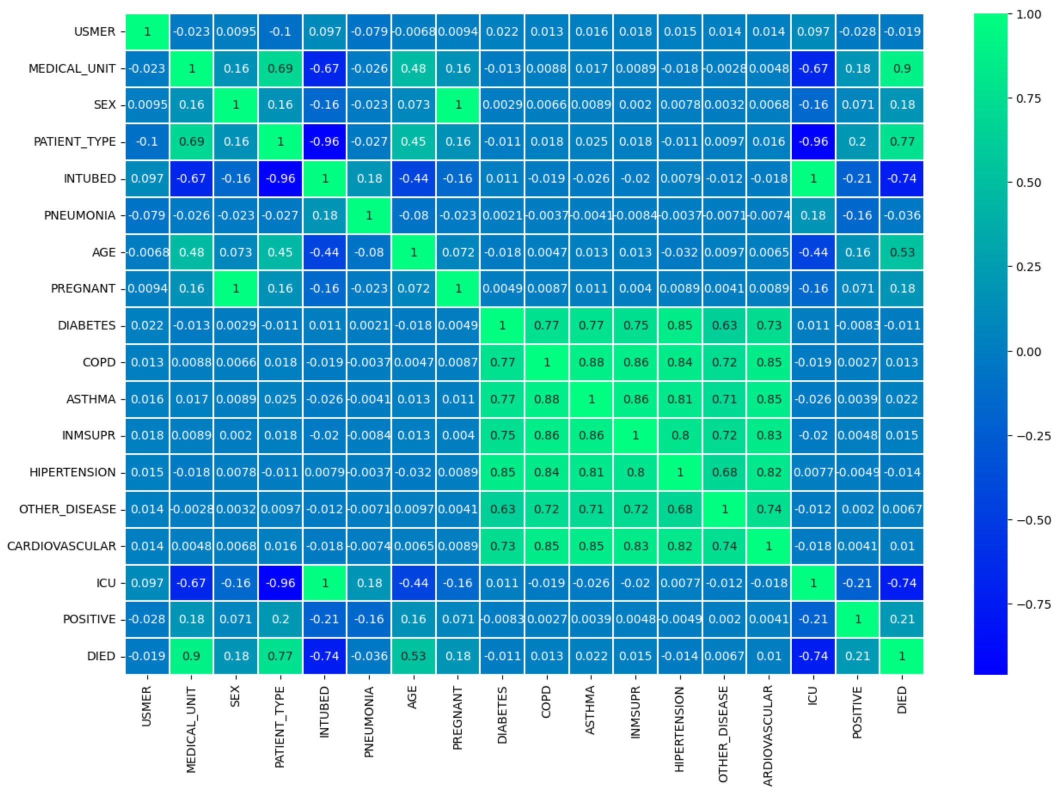


Fig. 5.7 Display feature selection

Figure 5.7 visualizes the comparison of the features that are important with respect to target feature using corelation heat map.

5.1.1 BEFORE FEATURE SELECTION

The results before feature selection are as follows:

a. Stochastic gradient descent:

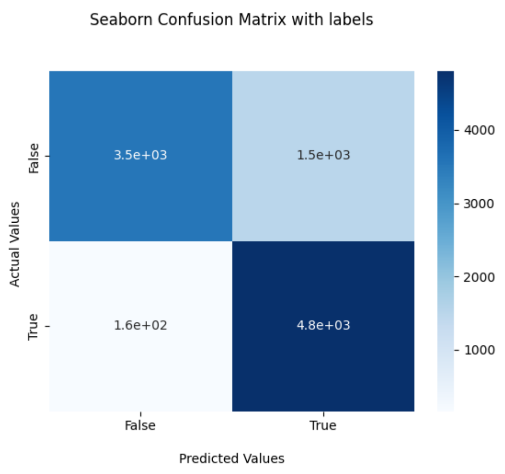


Fig 5.8 (a) Confusion matrix of SGD (Before Feature Selection)

b. Random Forest

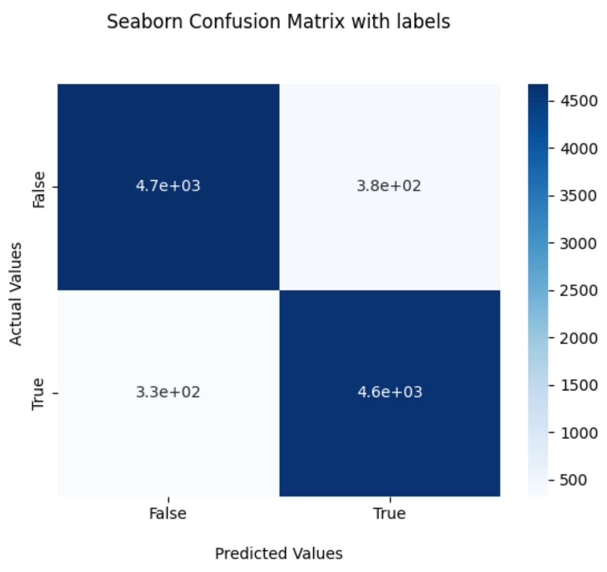


Fig 5.9 (b) Confusion matrix of RF (Before Feature Selection)

c. XGBoost

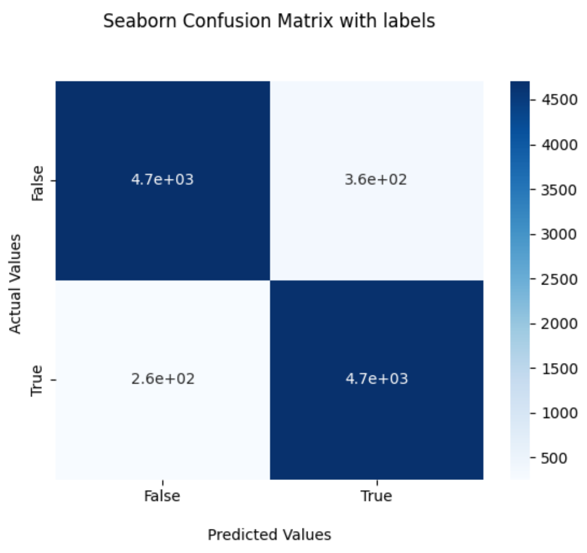


Fig 5.10 (c) Confusion matrix of XGBoost (Before Feature Selection)

d. K-Nearest Neighbour

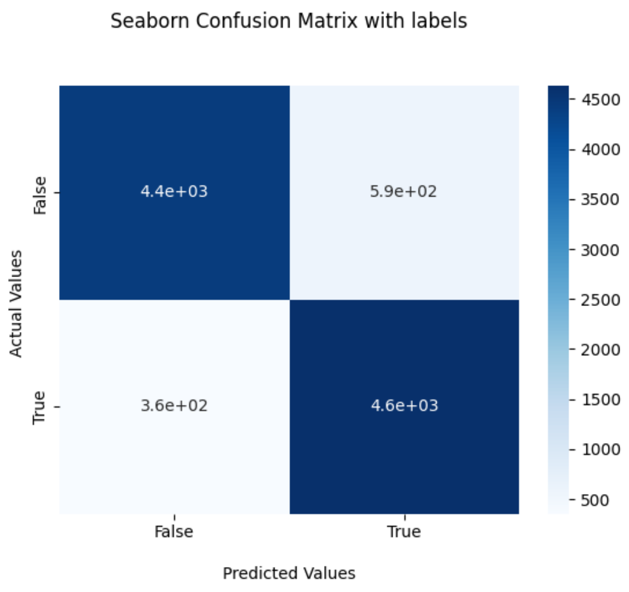


Fig 5.11 (d) Confusion matrix of KNN (Before Feature Selection)

e. Gradient Boosting

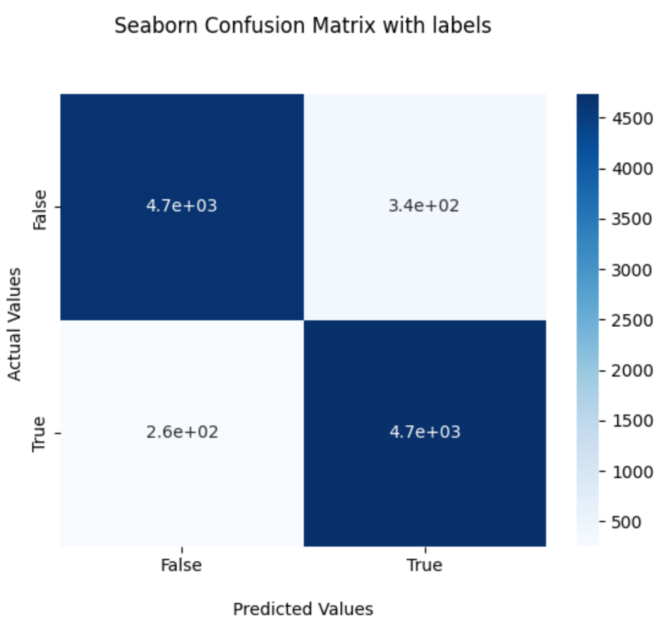


Fig 5.12 (e) Confusion matrix of GB (Before Feature Selection)

f. Decision Tree

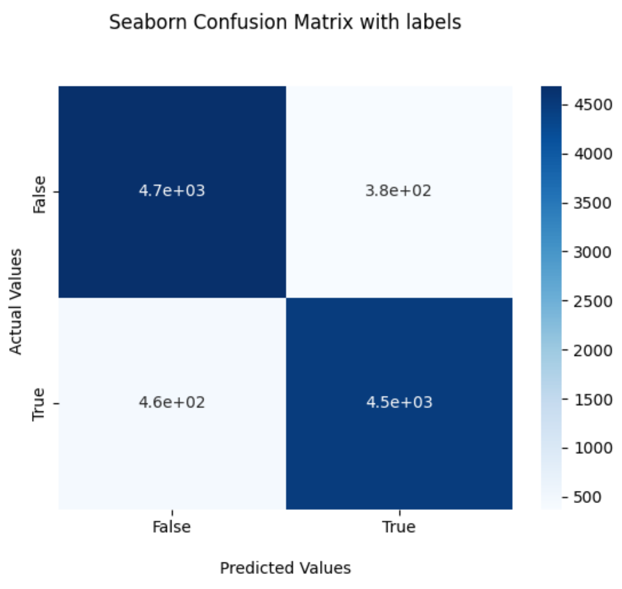


Fig 5.13 (f) Confusion matrix of DT (Before Feature Selection)

g. Neural Network

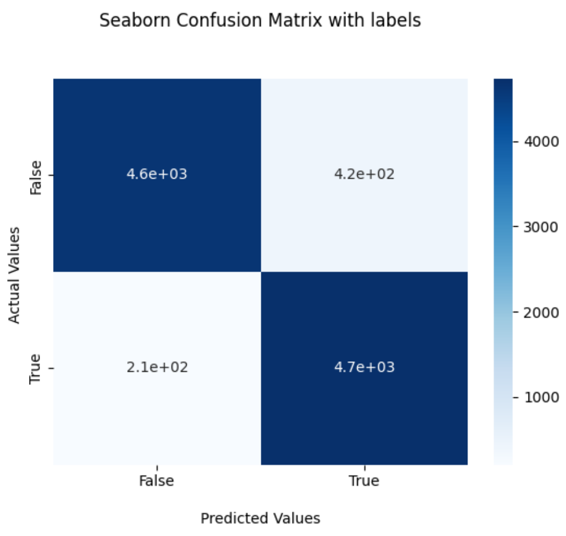


Fig 5.14 (g) Confusion matrix of NN (Before Feature Selection)

5.1.2 AFTER FEATURE SELECTION

The results before feature selection are as follows:

a. Stochastic Gradient Descent

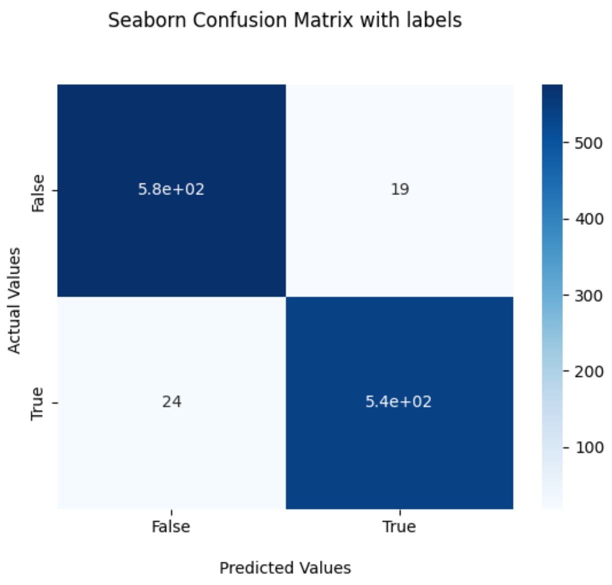


Fig 5.15 (a) Confusion matrix of SGD (After Feature Selection)

b. Random Forest

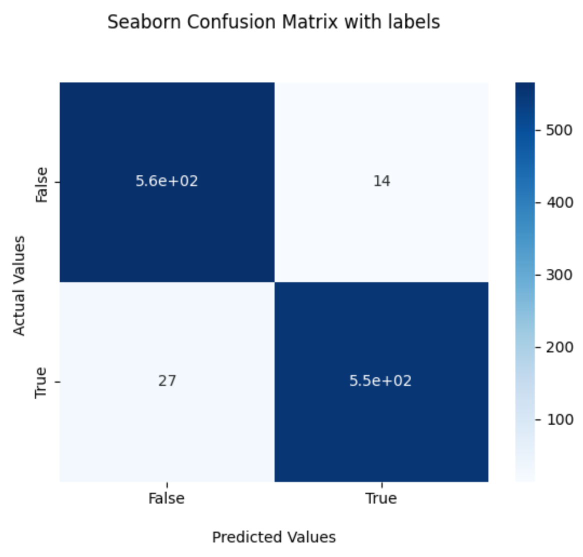


Fig 5.16 (b) Confusion matrix of RF (After Feature Selection)

c. XGBoost

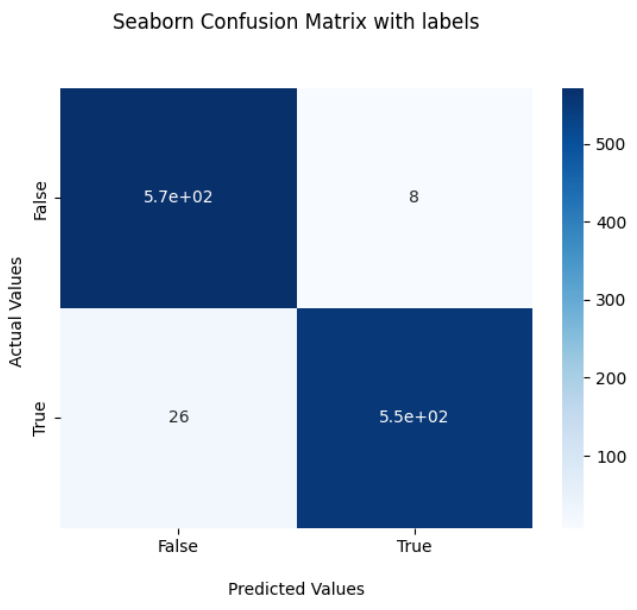


Fig 5.17 (c) Confusion matrix of XGBoost (After Feature Selection)

d. K-Nearest Neighbour

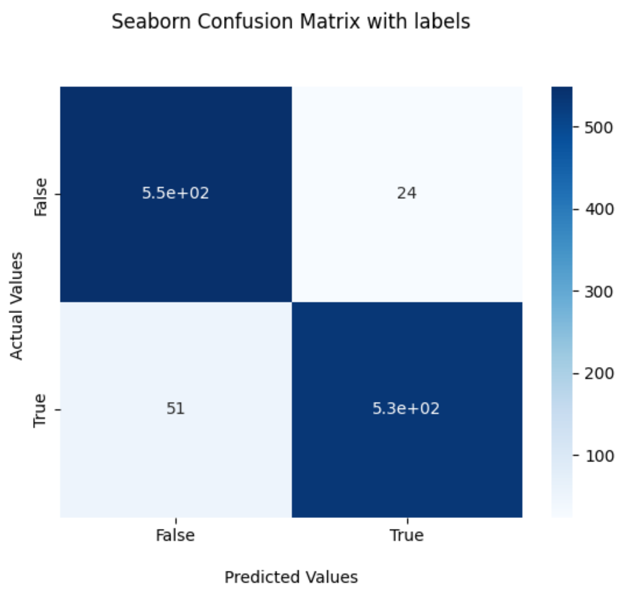


Fig 5.18 (d) Confusion matrix of KNN (After Feature Selection)

e. Gradient Boosting

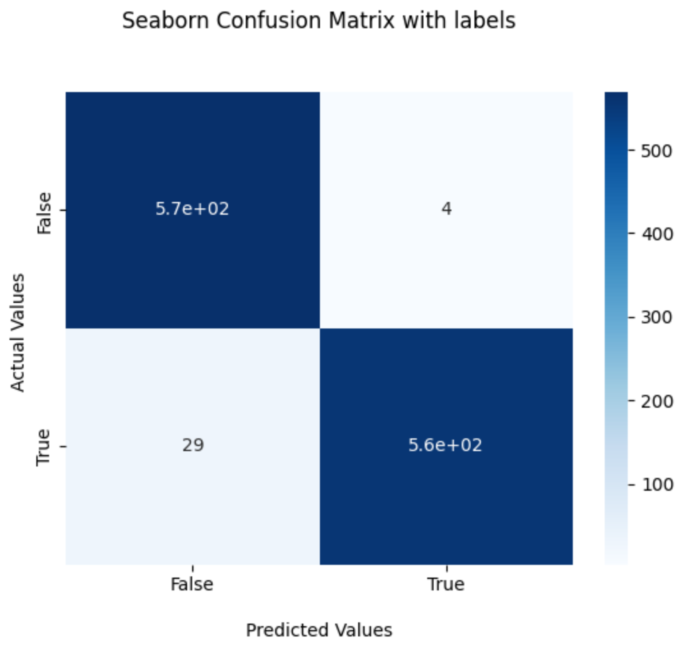


Fig 5.19 (e) Confusion matrix of GB (After Feature Selection)

f. Decision Tree

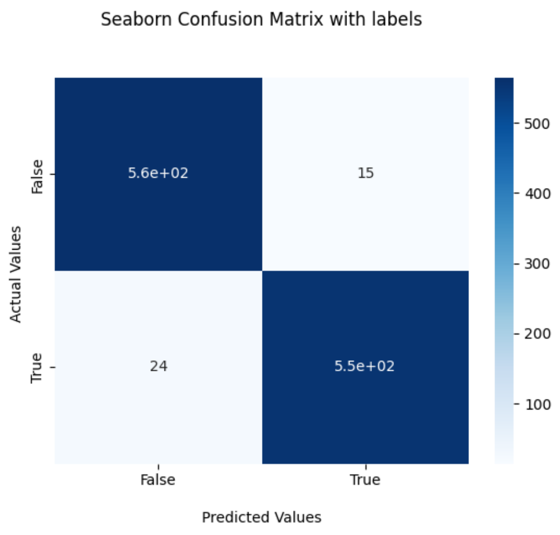


Fig 5.20 (f) Confusion matrix of DT (After Feature Selection)

g. Neural Network

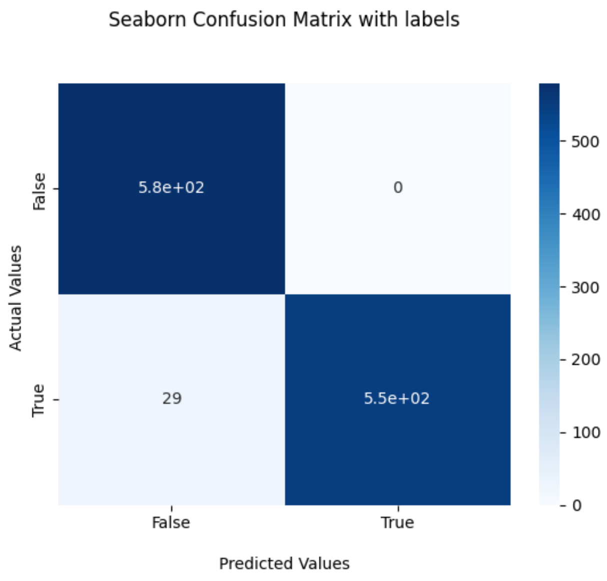


Fig 5.21 (g) Confusion matrix of NN (After Feature Selection)

5.1.3 OVERALL RESULTS

In this overall analysis of results is done.All seven base classifiers are use for the prediction and classification of death and recovered cases.

Table 5.1 describes the classification results before feature selection and Neurak Networks outperforms from all other base classifiers and figure 2.22 is the comparision of base classifiers.

Table 5.1 Classification results using base classifier before Feature Selection

Base Classifier	Accuracy(%)	Precission	F1-score	MCC
SGD	83.46	0.76	0.83	0.69
Random Forest	85.86	0.89	0.9	0.87
XGBoost	87.44	0.9	0.91	0.87
KNN	84.11	0.88	0.88	0.81
Gradient Boosting	90.45	0.92	0.93	0.88
Decision Tree	86.32	0.89	0.89	0.93
Neural Network	92.58	0.91	0.94	0.87

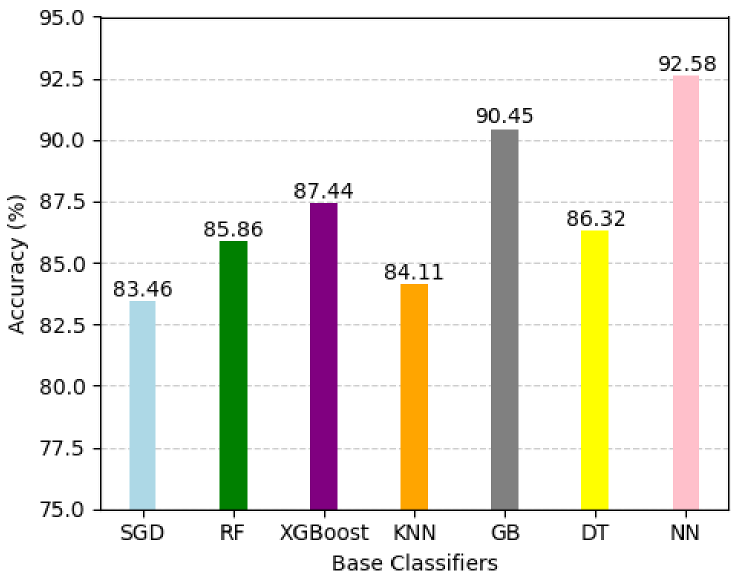


Fig. 5.22 Comparison of accuracy without hyperparameter tunning and before feature selection

Table 5.2 Classification results using base classifier without hyperparameter tunning and after feature selection

Base Classifier	Accuracy(%)	Precision	F1-score	MCC
SGD	97.06	0.98	0.97	0.94
Random Forest	96.54	0.97	0.97	0.93
XGBoost	97.06	0.98	0.97	0.94
KNN	93.51	0.95	0.94	0.87
Gradient Boosting	97.14	0.99	0.97	0.94
Decision Tree	96.62	0.97	0.97	0.93
Neural Network	97.49	1	0.97	0.95

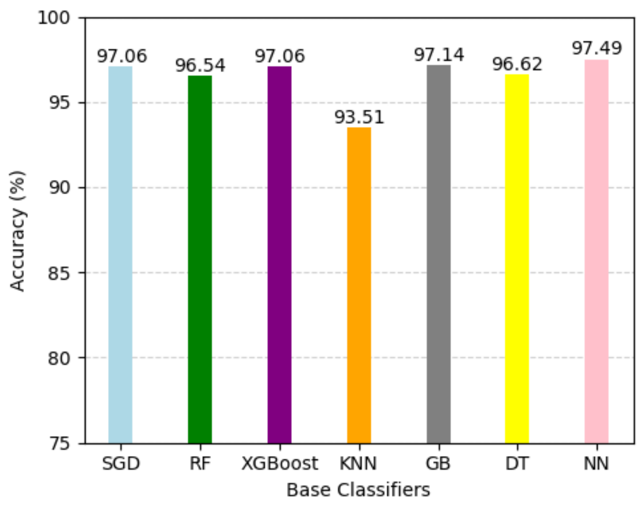


Fig. 5.23 Comparison of accuracy without hyperparameter tunning and after feature selection

Table 5.3 Classification results using base classifier with hyperparameter tunning and after feature selection

Base Classifier	Accuracy(%)	Precission	F1-score	MCC
SGD	97.32	0.97	0.97	0.94
Random Forest	97.75	0.98	0.98	0.95
XGBoost	97.66	0.98	0.98	0.95
KNN	95.85	0.96	0.96	0.91
Gradient Boosting	97.75	0.98	0.98	0.95
Decision Tree	97.66	0.98	0.98	0.95
Neural Network	97.85	1	0.97	0.95

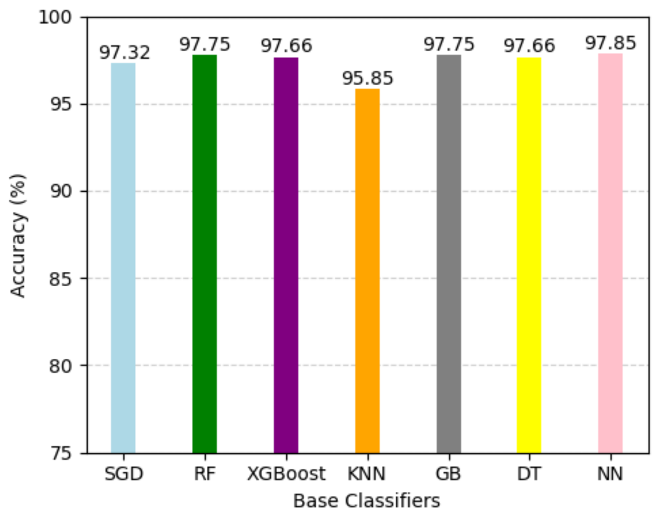


Fig. 5.24 Comparison of total accuracy with hyper parameter tunning and after feature selection

Table 5.2 and 5.3 describes the classification results after feature selection and with and without hyperparameter tunning and Neurak Networks outperforms from all other base classifiers and figure 5.23 and 24 gives the comparision of base classifiers.

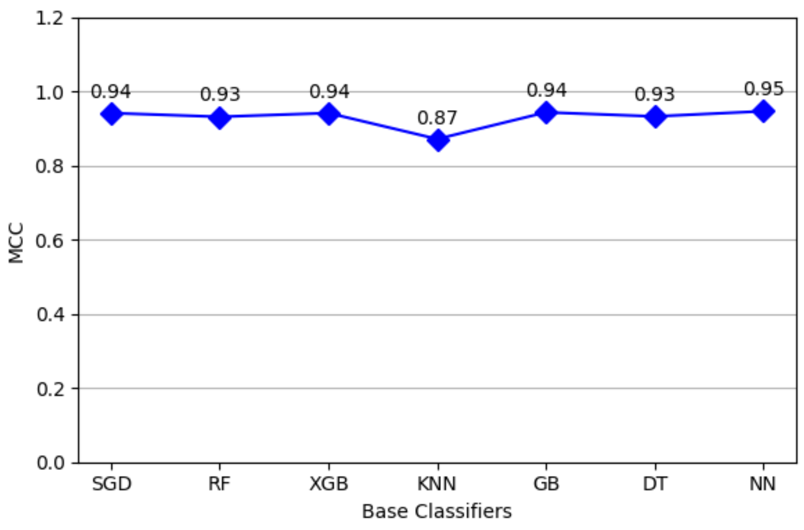


Fig.5.25 Calculation of MCC without hyper parameter tunning

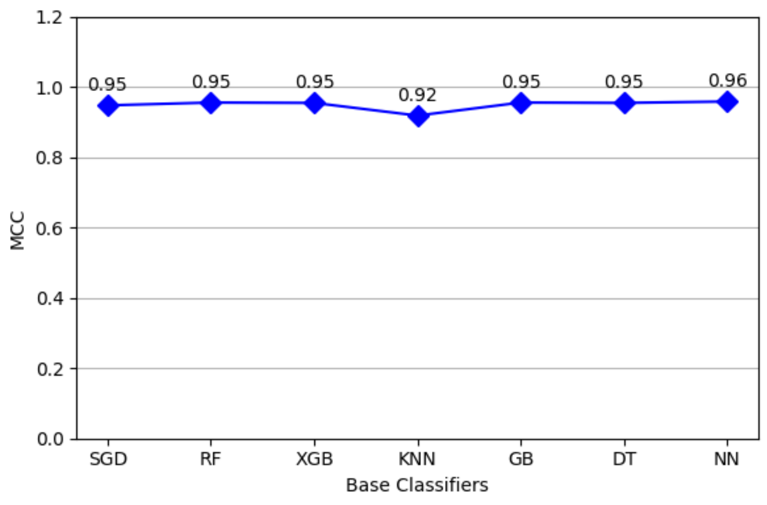


Fig.5.26 Calculation of MCC with hyper parameter tunning

Fig 5.25 and 26 Describes the comparison of MCC with different base classifiers applied with and without hyperparameter tunning and Neural Network is having the maximum MCC as compared to all others.

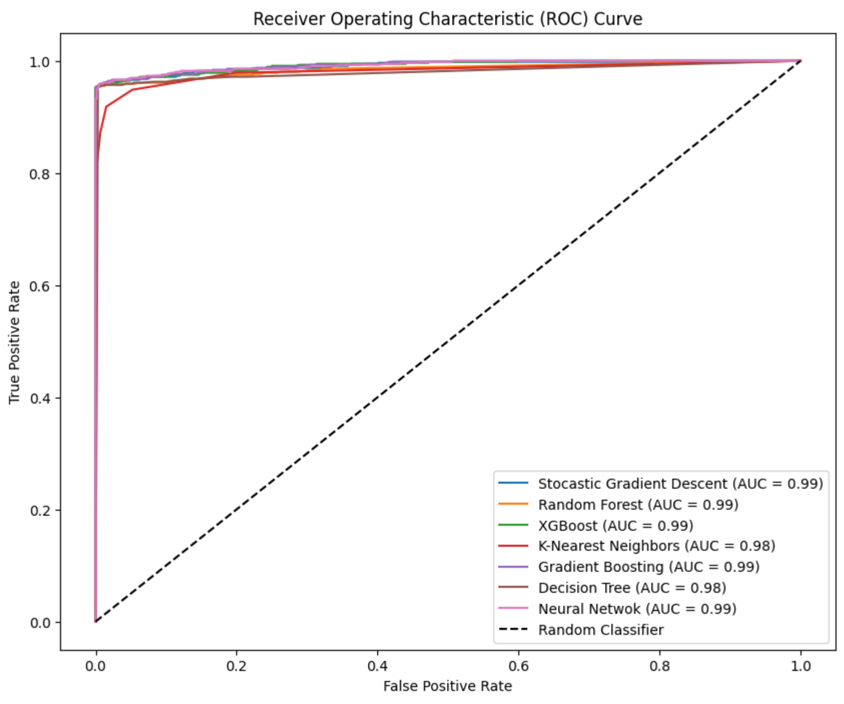


Fig. 5.27 Calculation of ROC curve

Figure 5.27 Describes the ROC curve of all the seven base classifiers used.

CHAPTER 6: CONCLUSIONS AND FUTURE SCOPE

6.1 CONCLUSION

In past covid-19 has witnessed a dramatic increase in the cases. This project introduces various machine learning models to classify and predict death and recovered cases. The results demonstrates that before feature selection Neural Network model plays a vital role in the classification and prediction of covid – 19 death and recovered cases as it produces an accuracy of 92.58%. Then after feature selection also Neural Network model plays a vital role in the classification and prediction of covid – 19 death and recovered cases as it produces more accurate results compared to other models after that Gradient Boosting also achieved the best accuracy as compared to other classifiers. The Neural Network achieve high accuracy of 97.49 % without hyperparameter tunning and with hyperparameter tunning it achieves an accuracy of 97.85 % respectively.

Our project has application in various fields like in public health intervention it can provide insights about the impact of different symptoms to prevent the impact on the persons health. Also, it provides some epidemiological studies that can aid researchers to understand the disease dynamics and also identify the patterns. The treatment planning can also be identified by understanding the likelihood and the recovered and death cases for every patient and can aid the healthcare professionals in developing personalized treatment plans.

6.2 FUTURE SCOPE

We will apply more advanced methods using machine learning as well to make this project more effective and apply certain new techniques that are emerging in this field.

REFERENCES

[1] Tarun Sai Lomte (2023, Mar. 14). Researchers model possible COVID-19 trajectories .Available at: https://www.news-medical.net/news/20230314/Researchers-model-possible-COVID-19-trajectories-for-2023.aspx.

[2] Ihme (2022, Dec. 01). Covid-19 projections Institute for Health Metrics and Evaluation. Available at: https://covid19.healthdata.org/india.

[3] S. Phillips (2022, Sep. 02). Prediction markets and the future of covid-19, STAT. Available at: https://www.statnews.com//prediction-markets-and-the-future-of-covid-19.

[4] R.L. Kumar, F. Khan, S. Din, S.S. Band, A. Mosavi and E. Ibeke, “Recurrent neural network and reinforcement learning model for COVID-19 prediction”. Frontiers in public health, 9, p.744100, 2021.

[5] R. Sujath, J.M. Chatterjee and A.E. Hassanien, “A machine learning forecasting model for COVID-19 pandemic in India”, In Stochastic Environmental Research and Risk Assessment, 34(7), pp. 959–972. doi:10.1007/s00477-020-01827-8, 2020.

[6] H. Aslam and S. Biswas, “Analysis of COVID-19 Death Cases Using Machine Learning”, SN Computer Science, 4(4), p.403, 2023.

[7] M.O Alassafi, M. Jarrah and R. Alotaibi, “Time series predicting of COVID-19 based on deep learning.” Neurocomputing, 468, pp.335-344, 2022.

[8] R. Kumari, S. Kumar, R.C. Poonia, V. Singh, L. Raja, V. Bhatnagar and P. Agarwal, “Analysis and predictions of spread, recovery, and death caused by COVID-19 in India” ,In Big Data Mining and Analytics, 4(2), pp.65-75, 2021.

[9] V.K. Gupta, A. Gupta, D. Kumar and A. Sardana, “Prediction of COVID-19 confirmed, death, and cured cases in India using random forest model.”, In Big Data Mining and Analytics, 4(2), pp.116-123, 2021.

[10] M. Alazab, A. Awajan, A. Mesleh, A. Abraham, V. Jatanaand, S. Alhyari, “COVID-19 prediction and detection using deep learning”, In International Journal of Computer Information Systems and Industrial Management Applications, pp.168-18, 2020.

[11] A. Zeroual, F. Harrou, A. Dairi and Y. Sun, “Deep learning methods for forecasting COVID-19 time-Series data: A Comparative study”, In Chaos, solitons & fractals, 140, p.110121, 2020.

[12] P. Wang, X. Zheng, J. Li and B. Zhu, "Prediction of epidemic trends in COVID-19 with logistic model and machine learning technics", In Chaos, Solitons & Fractals, 139, p.110058, 2020.

[13] V. Bhadana, A.S. Jalal and P. Pathak, "A comparative study of machine learning models for COVID-19 prediction in India", In IEEE 4th conference on information & communication technology (CICT) (pp. 1-7), 2020.

[14] C. An, H. Lim, D.W. Kim, J.H. Chang, Y.J. Choi and S.W. Kim, "Machine learning prediction for mortality of patients diagnosed with COVID-19: a nationwide Korean cohort study", Scientific reports, 10(1), p.18716, 2020

[15] S. Shastri, K. Singh, S.Kumar, P. Kourand V. Mansotra, 2020. Time series forecasting of Covid-19 using deep learning models: India-USA comparative case study. Chaos, Solitons & Fractals, 140, p.110227.

[16] S.S. Aljameel, I.U. Khan, N. Aslam, M. Aljabri and E.S. Alsulmi, "Machine learning-based model to predict the disease severity and outcome in COVID-19 patients". In Scientific Programming, pp.1-10, 2021.

[17] E. Fayyoumi, S. Idwan and H. AboShindi, "Machine learning and statistical modelling for prediction of novel COVID-19 patients case study: Jordan", In International Journal of Advanced Computer Science and Applications, 11(5), 2020.

[18] S.I. Amari, Backpropagation and stochastic gradient descent method, "Neurocomputing". pp.185-196,1993.

[19] C. Vens and F. Costa "Random forest-based feature induction", In IEEE 11th International Conference on Data Mining (pp.744-753), 2011.

[20] T. Chenand and C. Guestrin, "Xgboost: A scalable tree boosting system", In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794), 2016.

[21] S. Zhang, X. Li, M. Zong, X. Zhu and R. Wan, "Efficient KNN classification with different numbers of nearest neighbors", In IEEE transactions on neural networks and learning systems, 29(5), pp.1774-1785, 2017.

[22] A. Natekin and A. knoll, "Gradient boosting machines, a tutorial", In Frontiers in neurorobotics, 7, p.21, 2013.

[23] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation", In BMC genomics, 21(1), pp.1-13, 2020.

[24] V.R. Basili and R.W. Selby, “Comparing the effectiveness of software testing strategies”, In IEEE transactions on software engineering, (12), pp.1278-1296, 1987.

[25] M.W. Browne, “Cross-validation methods”, In Journal of mathematical psychology, 44(1), pp.108-132, 2000.

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ....

Type of Document (Tick): **PhD Thesis** **M.Tech Dissertation/ Report** **B.Tech Project Report** **Paper**

Name: ______________________________**Department:** _________________**Enrolment No** _________

Contact No. ______________________________**E-mail.** ______________________________________

Name of the Supervisor: ____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _________________________

____

____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

– Total No. of Pages =
– Total No. of Preliminary pages =
– Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at.................. (%). Therefore, we
are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

Covid-19

ORIGINALITY REPORT

9%	7%	6%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.ncbi.nlm.nih.gov Internet Source	1%
2	media.neliti.com Internet Source	<1%
3	Abdelhafid Zeroual, Fouzi Harrou, Abdelkader Dairi, Ying Sun. "Deep Learning Methods for Forecasting COVID-19 Time-Series Data: A Comparative Study", Chaos, Solitons & Fractals, 2020 Publication	<1%
4	dc.tsinghuajournals.com Internet Source	<1%
5	Sourabh Shastri, Kuljeet Singh, Sachin Kumar, Paramjit Kour, Vibhakar Mansotra. "Time series forecasting of Covid-19 using deep learning models: India-USA comparative case study", Chaos, Solitons & Fractals, 2020 Publication	<1%
6	Submitted to Indian School of Business Student Paper	<1%

7	www.frontiersin.org Internet Source	<1 %
8	www.ijisrt.com Internet Source	<1 %
9	www.sciencegate.app Internet Source	<1 %
10	ijrar.org Internet Source	<1 %
11	Dmytro Chumachenko, Tetiana Dudkina, Sergiy Yakovlev, Tetyana Chumachenko. "Effective Utilization of Data for Predicting COVID-19 Dynamics: An Exploration through Machine Learning Models", International Journal of Telemedicine and Applications, 2023 Publication	<1 %
12	Submitted to Cranfield University Student Paper	<1 %
13	Soham P. Chinchalkar, Rachna K. Somkunwar. "An Innovative Keylogger Detection System Using Machine Learning Algorithms and Dendritic Cell Algorithm", Revue d'Intelligence Artificielle, 2024 Publication	<1 %
14	arxiv.org Internet Source	<1 %

15	bmcmedinformdecismak.biomedcentral.com Internet Source	<1%
16	www.kdnuggets.com Internet Source	<1%
17	Submitted to Liverpool John Moores University Student Paper	<1%
18	Submitted to The University of the West of Scotland Student Paper	<1%
19	pdfcoffee.com Internet Source	<1%
20	Submitted to City of Bath College, Avon Student Paper	<1%
21	www.banqueducanada.ca Internet Source	<1%
22	Firuz Kamalov, Khairan Rajab, Aswani Cherukuri, Ashraf Elnagar, Murodbek Safaraliev. "Deep Learning for Covid-19 Forecasting: state-of-the-art review.", Neurocomputing, 2022 Publication	<1%
23	"Pan-African Conference on Artificial Intelligence", Springer Science and Business Media LLC, 2023 Publication	<1%

24	dokumen.pub Internet Source	<1 %
25	Alaa Darabseh, Doyel Pal. "Performance Analysis of Keystroke Dynamics Using Classification Algorithms", 2020 3rd International Conference on Information and Computer Technologies (ICICT), 2020 Publication	<1 %
26	www.techscience.com Internet Source	<1 %
27	Vishan Kumar Gupta, Avdhesh Gupta, Dinesh Kumar, Anjali Sardana. "Prediction of COVID-19 confirmed, death, and cured cases in India using random forest model", Big Data Mining and Analytics, 2021 Publication	<1 %
28	fdocuments.in Internet Source	<1 %
29	researchrepository.wvu.edu Internet Source	<1 %
30	www.ijraset.com Internet Source	<1 %
31	"Communication and Intelligent Systems", Springer Science and Business Media LLC, 2024 Publication	<1 %

32	Nikhil Kumar, Naveen Kumar Shah, Virender Ranga. "Implementation of a mobile application for managing academic life of new college students", AIP Publishing, 2024 Publication	<1 %
33	deepai.org Internet Source	<1 %
34	docplayer.net Internet Source	<1 %
35	librarianreservecorps.libanswers.com Internet Source	<1 %
36	Madhvan Bajaj, Priyanshu Rawat, Vikrant Sharma, Satvik Vats, Satya Prakash Yadav, Vinay Kukreja. "Study on Degenerative Parkinson's Disease Using Various Machine Learning Algorithms", 2023 3rd Asian Conference on Innovation in Technology (ASIANCON), 2023 Publication	<1 %
37	Swapna rekha Hanumanthu. "Role of Intelligent Computing in COVID-19 Prognosis: A State-of-the-Art Review", Chaos, Solitons & Fractals, 2020 Publication	<1 %
38	www.coursehero.com Internet Source	<1 %