

CHILD MORTALITY PREDICTION USING MACHINE LEARNING

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

SAMIREDDY ADITHYA(Reg.No.38110493)

TUDIMALADINNA SANJAY REZINALD(Reg.No.38110603)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

JEPPIAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600 119

MARCH – 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **SAMIREDDY ADITHYA (Reg.No:38110493)** and **TUDIMALADINNA SANJAY REZINALD (Reg.No:38110603)** who carried out the project entitled "**CHILD MORTALITY PREDICTION USING MACHINE LEARNING**" under our supervision from **August 2021 to March 2022**

Internal Guide

Dr. A. VIJI AMUTHA MARY, M.Tech., Ph.D

Head of the Department

Dr. L. LAKSHMANAN M.E., Ph.D.,

Dr. S. VIGNESHWARI, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **SAMIREDDY ADITHYA (Reg.No:38110493)** and **TUDIMALADINNA SANJAY REZINALD (Reg.No:38110603)** hereby declare that the Project Report entitled “**CHILD MORTALITY PREDICTION USING MACHINE LEARNING**” done by me under the guidance of **Dr. A. Viji Amutha Mary, M.Tech., Ph.D**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfilment of the requirements for the award of **Bachelor of Engineering (B.E)** degree in **Computer Science and Engineering (CSE)**

DATE: 04-03-2022

**SAMIREDDY ADITHYA
TUDIMALADINNA SANJAY REZINALD**

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph.D., Dean**, School of Computing and **Dr. S. VIGNESWARI, M.E., Ph.D., and Dr. L. LAKSHMANAN, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. A. VIJI AMUTHA MARY, M.Tech., Ph.D** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of **Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Children Mortality refers to mortality of children under the age of five. The child mortality rate, also under-five mortality rate, refers to the probability of dying between birth and exactly five years of age. The mortality of children also happens in fetus. The aim is to investigate machine learning based techniques for classification of mortality fetal health classification results in best accuracy. The analysis of dataset by supervised machine learning technique (SMLT) to capture several information's like, variable identification, uni-variate analysis, bi-variate and multi-variate analysis, missing value treatments and analyze the data validation, data cleaning/preparing and data visualization will be done on the entire given dataset. Our analysis provides a comprehensive guide to sensitivity analysis of model parameters with regard to performance in the classification of fetal health. To propose a machine learning-based and additionally, to compare and discuss the performance of various machine learning algorithms for the given dataset.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	V
	LIST OF FIGURES	Viii
	LIST OF ABBREVIATIONS	viii
1.	INTRODUCTION	1
	MODEL IDE	3
	OBJECTIVE	3
2.	LITERATURE SURVEY	5
3.	AIM AND SCOPE OF THE	7
	PRESENT INVESTIGATION	
	SCOPE OF THE PROJECT	7
	SYSTEM ARCHITECTURE	7
	WORKFLOW DIAGRAM	9
	EXISTING SYSTEM	9
	PROPOSED SYSTEM	10
	SYSTEM REQUIREMENTS	10
4.	METHODOLOGY AND	11
	ALGORITHMS USED	
	INTRODUCTION TO MACHINE	11
	LEARNING	
	PACKAGES	14
	LIST OF MODULES USED	16
	ALGORITHMS USED IN	23

	THE PROJECT	
5.	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	33
6.	SUMMARY AND CONCLUSIONS	37
	REFERENCES	38
	APPENDIX	
A.	SOURCE CODE	
B.	SCREENSHOTS	
C.	PUBLICATION WITH PLAGARISM REPORT	

FIGURE.NO	NAME OF THE FIGURE	PAGE.NO
-----------	--------------------	---------

LIST OF FIGURES

1.	SYSTEM ARCHITECTURE	07
2.	WORK FLOW DIAGRAM	09

LIST OF ABBREVIATIONS

AI	-	ARTIFICIAL INTELLIGENCE
ML	-	MACHINE LEARNING
KNN	-	K-NEAREST NEIGHBOUR

CHAPTER-1

INTRODUCTION

Artificial intelligence (AI) is the study of how to get computers to do things on their own, without being explicitly programmed to do so. In the last decade, AI has brought us self-driving cars, feasible discourse acknowledgment, strong web search, and a vastly improved understanding of the human genome. AI is so pervasive today that you probably use it without even realising it on a daily basis. There are many experts who believe this is the greatest way to progress toward human-level AI. In this course, you will learn about the most effective AI techniques and practise putting them into action and getting them to function on their own. Furthermore, you'll study about the theoretical foundations of learning while also developing the practical skills necessary to apply these tactics to new situations. Finally, you'll learn about some of Silicon Valley's best practises for AI and AI development.

Support Vector Machines, Parts, and Neural Organizations. Learning how to do things on one's own (bunching, dimensionality decrease, recommender frameworks, profound learning). AI best practises (predisposition/change hypothesis; progress in AI and AI process). Additionally, you'll discover how to apply learning computations to the construction of clever robots (knowledge, control), text understanding (web search, spam threat), PC vision/clinical informatics/sound/information index mining and numerous domains.

Child mortality or the under-five mortality rate refers to the probability of a child dying between birth and exactly 5 years of age, expressed per 1,000 live births. In 2020, 5.0 million children under 5 years of age died. This translates to 13,800 children under the age of 5 dying every day in 2020. Globally, infectious diseases, including pneumonia, diarrhoea and malaria, remain a leading cause of under-five deaths, along with preterm birth and intrapartum-related complications.

The global under-five mortality rate declined by 61 per cent, from 93 deaths per 1,000 live births in 1990 to 37 in 2020. Despite this considerable progress, improving child survival remains a matter of urgent concern. In 2020 alone, roughly 13,800 under-five deaths occurred every day, an intolerably high number of largely preventable child deaths.

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect. The learning process is automated and improved based on the experiences of the machines throughout the process. Good quality data is fed to the machines, and different algorithms are used to build ML models to train the machines on this data. The choice of algorithm depends on the type of data at hand, and the type of activity that needs to be automated.

Machine Learning is widely used in the healthcare industry. It helps healthcare researchers to analyze data points and suggest outcomes. Natural language processing helped to give accurate insights for better results of patients. Further, machine learning has improved the treatment methods by analyzing external data on patients' conditions in terms of X-ray, Ultrasound, CT-scan, etc. NLP, medical imaging, and genetic information are key areas of machine learning that improve the diagnosis, detection, and prediction system in the healthcare sector.

Easily identifies trends and patterns. Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right

products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them. No human intervention needed (automation). With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam. Continuous Improvement. As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster. Handling multi-dimensional and multi-variety data. Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments. Wide Applications. You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

1.1.MODEL IDE:

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The IDE was installed in Anaconda, an open-source distribution for the languages Python and R used to perform Data Science and Machine Learning. The IDE's UI in which the model was developed.

1.2 OBJECTIVE:

Reduction of child mortality is reflected in several of the United Nations' Sustainable Development Goals and is a key indicator of human progress. Define a problem

The UN expects that by 2030, countries end preventable deaths of newborns and children under 5 years of age, with all countries aiming to reduce under 5 mortality to at least as low as 25 per 1,000 live births.

Parallel to notion of child mortality is of course maternal mortality, which accounts for 295 000 deaths during and following pregnancy and childbirth (as of 2017). The vast majority of these deaths (94%) occurred in low-resource settings, and most could have been prevented.

In light of what was mentioned above, Cardiotocograms (CTGs) are a simple and cost accessible option to assess fetal health, allowing healthcare professionals to take action in order to prevent child and maternal mortality. The equipment itself works by sending ultrasound pulses and reading its response, thus shedding light on fetal heart rate (FHR), fetal movements, uterine contractions and more.

CHAPTER-2

LITERATURE SURVEY

Using Public Health Data from Sao Paulo, Brazil, Machine Learning to Predict Neonatal MortalityLuciana Correia Alves.The mortality rate of newborn children is a major financial and well-being quality indicator. It is estimated that 70% of newborn deaths in Brazil are due to neonatal deaths. Despite its importance, newborn mortality is on the rise, raising concerns about the need for effective methods that can assist reduce this incidence of death. In this study, a new approach is offered to categorise infants at high risk of neonatal mortality by utilising AI algorithms to monitor their overall health. 15 858 entries from the SINASC and SIM databases in Sao Paulo city (Brazil) from the SP Neo Death dataset are used to illustrate the concept in this paper. As a result, an average AUC of 0.96 was achieved while using SVM, XGBoost, Logistic Regression, and Random Forests AI algorithms to classify tests as either defenceless to death or not. In addition, the SHAP technique was used to identify the factors that had the greatest impact on the final results of the calculations.Mortality and Morbidity. Anne Case claims that Angus Deaton The mortality and gloominess rates among white non-Hispanic Americans in their mid-fifties have been steadily rising since the turn of the century, according to our latest research (Case and Deaton 2015). More deaths from drug overdoses, suicides, and liver disease caused by alcohol are to blame for the rise in the overall mortality rate among whites, especially among those with only a high school diploma or less. Among white non-Hispanics (people and women), mortality rises for those without a general preparation, whereas it declines for those who have a master's degree in education. Death rates among blacks and Hispanics have continued to reduce, despite the fact that educational achievements have been made in this area. It used to be that death rates in the United States were depicted as high as they are in other wealthy

countries. The results of helpless death cannot be fully explained by current levels of assets, particularly those that are slowly growing old, stale, and, in any case, diminishing. While it is possible that this is just a beginning point, we believe it is a plausible scenario in which the deterioration of employment opportunities for whites with low levels of education is a catalyst for a chain reaction that affects the lives of both parents and their children. An important piece of data is shown to have terrible effects, even if the methodology in question is successful in promoting pay. Youngster mortality and richness decrease According to Matthias Doepke. According to Barro and Becker, children's mortality and fruitfulness are linked in three distinct ways. "The ripeness choice is constant in the pattern model, and the number of enduring children is not a vulnerability. Stochastic mortality and an arrangement with consecutive fruitfulness decisions distinguish the pattern model, which is expanded from discrete ripeness decision. The models' quantitative forecasts are quite similar. Despite the fact that the hard and fast ready rate declines in each model due to an increase in child mortality, the number of suffering young people increases. Net generation rates in industrialised nations have dropped dramatically over the previous century due to a variety of variables other than decreased infant and youngster mortality.. Analysis and Prediction of Child Mortality in India According to Rushikesh Gawande. One of the responsibilities of progress coordinators and pioneers is to lower the mortality rate and broaden the scope of prosperity care. There must be a breakdown of the death rates and the reasons for the variability in order to come up with effective strategies. There are several factors that influence the rate of child mortality, such as the rate of birth, the rate of foetuses born, the proficiency of the mother, and the level of undernourishment. Additionally, a thorough examination of diseases such as jungle fever, loose bowels, and others that affect newborns and their contribution to infant mortality is carried out. Keeping up good health and reducing death rates necessitates an analysis of PHCs that meet the requirements. Geographical analysis of

specific PHC locations can also be utilised to determine the demand for more PHCs and to predict their locations.

CHAPTER-3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

SCOPE OF THE PROJECT:

The progress in reducing child mortality around the world has been remarkable. Under-five mortality rates have declined by almost 60 per cent since 1990, and as a result millions more children survive to adolescence today than they did three decades ago. The impact of the ongoing COVID-19 pandemic, however, threatens years of improvement in child and adolescent survival through the interruption of essential health services. Even before the coronavirus captured the world's attention, it was clear that if survival targets were to be met, resources and policy would need to be geared towards accelerating progress and not just maintaining it.

SYSTEM ARCHITECTURE:

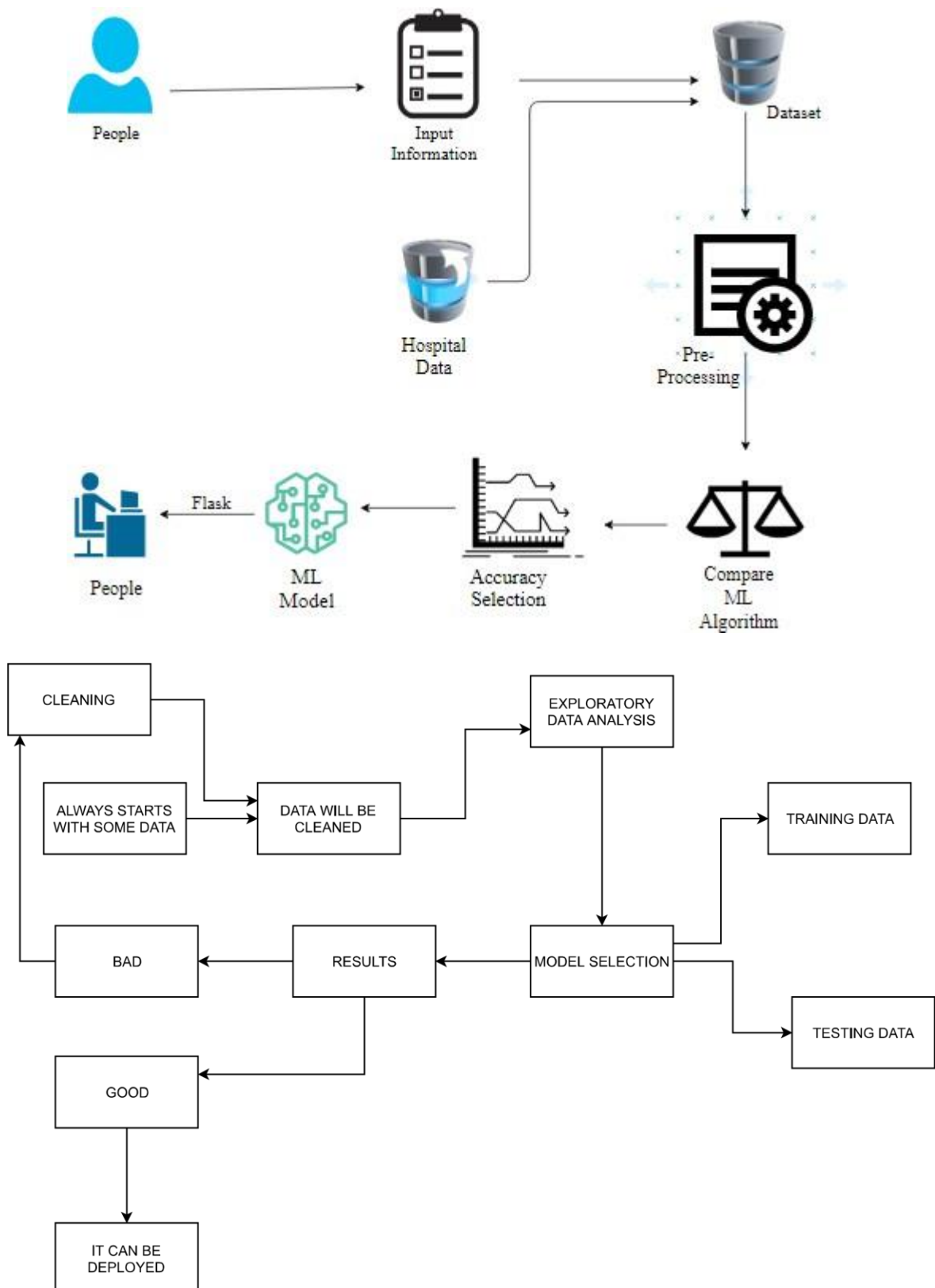


Fig-1

WORK FLOW DIAGRAM:

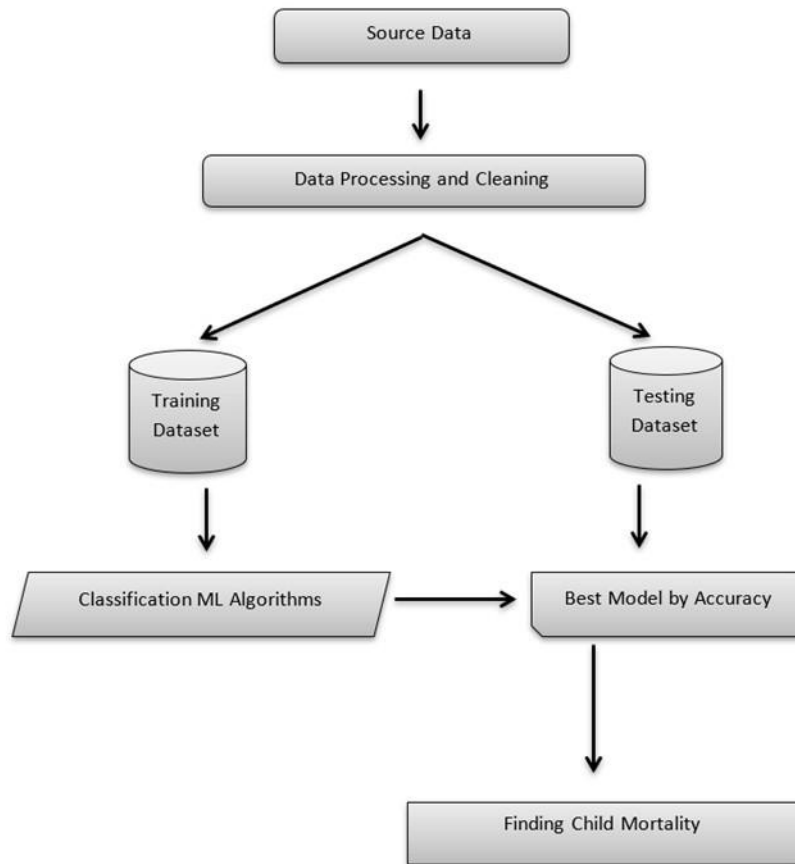


Fig-2

EXISTING SYSTEM:

They present a novel data-driven modeling approach for post-ICU mortality prediction based on the temporal dynamics of patient condition during ICU stay. Specifically, we develop a new switching state-space model that coverts and fuses various patient variables into a SAPS II-based sequence to represent patient condition. We further model the relationship of in-ICU patient condition and Post ICU patient survival using a logistic regression model. A close monitoring of the severity level over time during ICU stay is more preferable. In this study, we design a

new switching state-space model by correlating patient's condition dynamics in last hours of ICU stay to the risk probabilities in a short time period after ICU discharge. The training data with minority class (death) is oversampled to balance the data. In addition to predict mortality of ICU patients, we are also interested in discovering which physiological variables are best discriminated upon the predicted outcomes.

Disadvantages:

1. Only the ICU mortality has been calculated for which the data will be less.
2. The performance metric can be compared and a better prediction can be done.

PROPOSED SYSTEM:

The process can be automated and the machine learning algorithms are applied for better performance of the model.

The mortality rate can be analyzed properly and better prediction can be done.

Proposed work and the first the data pre-processing and data cleaning is done. by this process the data is filtered and now the data is inserted into algorithm .After that the 4 algorithms are to be compared to analyze the accuracy of child mortality . to find the accuracy percentage of time with abnormal long term variability , mean value of long term variability, histogram tendency and various variabilities and Tendencies are calculated. With these characteristics we found the accuracy.

SYSTEM REQUIREMENTS:

SOFTWARE REQUIREMENTS:

Operating System : Windows

Tool : Anaconda with Jupyter Notebook

HARDWARE REQUIREMENTS:

Processor : Pentium IV/III

Hard disk : minimum 80 GB

RAM : minimum 2 GB

CHAPTER-4

METHODOLOGY AND ALGORITHMS USED

INTRODUCTION TO MACHINE LEARNING:

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labeling to learn data has to be labeled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/

labels or categories. Classification predictive modeling is the task of

approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is $y = f(X)$. The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include logistic regression, multi-class classification, Decision Trees and support vector machines etc. Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. Supervised learning problems can be further grouped into Classification problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

Machine learning is a field of computer science that uses statistical techniques to give computer systems the ability

to "learn" (e.g., progressively improve performance on a specific task) with data, without being explicitly programmed. The name machine learning was coined in 1959 by Arthur Samuel. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data – such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions, through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult or unfeasible; example applications include email filtering, detection of network intruders, and computer vision. Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data mining, where the latter sub-field focuses more on exploratory data analysis and is known as unsupervised learning. Within the field of data analytics, machine learning is a method used to devise complex models and algorithms that lend themselves to prediction; in commercial use, this is known as predictive analytics. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable

decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data.

PACKAGES:

The packages used in this model include:

Pandas

NumPy

Scikit-learn

Scikit-plot

Matplotlib's pyplot

Seaborn

Pydot

Pandas

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal. o The two primary data structures of pandas, Series (1- dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, DataFrame provides everything that R's data.frame provides and much more. pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries. o Written in: Python, Cython and C

NumPy

NumPy is a library for Python, adding support for large, multidimensional arrays and matrices, along with a large collection of highlevel mathematical functions to operate on these arrays. NumPy is an open-source software and has many contributors. o NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops using NumPy. 14 o Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. o In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. o Written in: Python and C

MODEL BUILDING PACKAGE:

Scikit-learn

Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines(svm), random forest, gradient boosting, kmeans and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. o It has been built on top of NumPy, SciPy and matplotlib. o Written in: Python, Cython, C and C++

3.5.3 DATA VISUALIZATION PACKAGES:

Scikit-plot

Scikit-plot is the result of an unartistic data scientist's dreadful realization that visualization is one of the most crucial components in the data

science process, not just a mere afterthought. o 38 Gaining insights is simply a lot easier when you're looking at a colored heatmap of a confusion matrix complete with class labels rather than a single-line dump of numbers enclosed in brackets. Besides, if you ever need to present your results to someone 15 (virtually any time anybody hires you to do data science), you show them visualizations, not a bunch of numbers in Excel. o All in all, it is an intuitive library used to add plotting functionality to a scikit-learn object. o Written in: Python, Cython, C and C++.

Matplotlib's pyplot

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits. o It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt or GTK+. o `matplotlib.pyplot` provides a MATLAB-like plotting framework. o `pylab` combines `pyplot` with `numpy` into a single namespace. This is convenient for interactive work, but for programming it is recommended that the namespaces be kept separate. o Written in: Python

MODULES IN THE PROJECT:

Data Wrangling

In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. Make sure that the document steps carefully and justify for cleaning decisions.

Data collection

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set

and Test set. The Data Model which was created using Random Forest, logistic, Decision tree algorithms and Support vector classifier (SVC) are applied on the Training set and based on the test result accuracy, Test set prediction is done.

Preprocessing

The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be preprocessed so as to improve the efficiency of the algorithm. The outliers have to be removed and also variable conversion need to be done.

DATA PRE-PROCESSING

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to

understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

A number of different data cleaning tasks using Python's Pandas library and specifically, it focus on probably the biggest data cleaning task, missing values and it able to more quickly clean data. It wants to spend less time cleaning data, and more time exploring and modeling.

Some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It's important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how to deal with filling in the missing values and to detect missing values, and do some basic imputation and detailed statistical approach for dealing with missing data. Before, joint into code, it's important to understand the sources of missing data. Here are some typical reasons why data is missing:

- User forgot to fill in a field.
- Data was lost while transferring manually from a legacy database.
- There was a programming error.
- Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

Variable identification with Uni-variate, Bi-variate and Multi-variate analysis:

import libraries for access and functional purpose and read the given dataset

General Properties of Analyzing the given dataset

Display the given dataset in the form of data frame

show columns

shape of the data frame

To describe the data frame

Checking data type and information about dataset

Checking for duplicate data

Checking Missing values of data frame

Checking unique values of data frame

Checking count values of data frame

Rename and drop the given data frame

To specify the type of values

To create extra columns

MODULE DIAGRAM:



GIVEN INPUT EXPECTED OUTPUT

input : data

output : removing noisy data

DATA VALIDATION/ CLEANING/PREPARING PROCESS

Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning model's and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate

process. The steps and techniques for data cleaning will vary from dataset to dataset. The primary goal of data cleaning is to detect and remove errors and anomalies to increase the value of data in analytics and decision making.

EXPLORATION DATA ANALYSIS OF VISUALIZATION

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end.

Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data.

- ☐ How to chart time series data with line plots and categorical quantities with bar charts.
- ☐ How to summarize data distributions with histograms and box plots.

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : visualized data

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. To achieving better results from the applied model in Machine Learning method of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values. Therefore, to execute random forest algorithm null values have to be managed from the original raw data set. And another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in given dataset.

False Positives (FP): A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

False Negatives (FN): A person who default predicted as payer. When actual class is yes but predicted class in no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

True Positives (TP): A person who will not pay predicted as defaulter. These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

True Negatives (TN): A person who default predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

Comparing Algorithm with prediction in the form of best accuracy result

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning algorithms in Python with scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare. Each model will have different performance characteristics. Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data. It needs to be able to use these estimates to choose one or two best models from the suite of models that you have created. When have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize. A way to do this is to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.

ALGORITHMS USED:

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc. In Supervised Learning, algorithms learn from labeled data. After understanding the data, the algorithm determines which label should be given to new data based on pattern and associating the patterns to the unlabeled new data.

LOGISTIC REGRESSION

It is a statistical method for analysing a data set in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.).

In other words, the logistic regression model predicts $P(Y=1)$ as a function of X . Logistic regression Assumptions:

- ☐ Binary logistic regression requires the dependent variable to be binary.
- ☐ For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome.
- ☐ Only the meaningful variables should be included.
- ☐ The independent variables should be independent of each other. That is, the model should have little.
- ☐ The independent variables are linearly related to the log odds.
- ☐ Logistic regression requires quite large sample sizes.

```

get the mean of each row
print("")
print("Accuracy result of Logistic Regression is:", accuracy.mean() * 100)
LR=accuracy.mean() * 100

Classification report of Logistic Regression Results:

```

	precision	recall	f1-score	support
0	0.91	0.94	0.92	332
1	0.75	0.67	0.71	94
accuracy			0.88	426
macro avg	0.83	0.88	0.82	426
weighted avg	0.87	0.88	0.88	426

```

Confusion Matrix result of Logistic Regression is:
[[311  21]
 [ 31  63]]

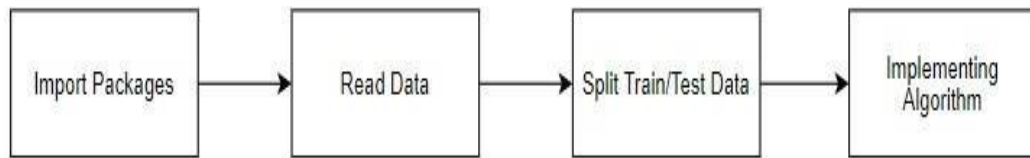
Sensitivity : 0.9367469879518072
Specificity : 0.6702127659574468

Cross validation test results of accuracy:
[0.88732394 0.86588235 0.89176471 0.87294118 0.73411765]

Accuracy result of Logistic Regression is: 85.04059652029825

```

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

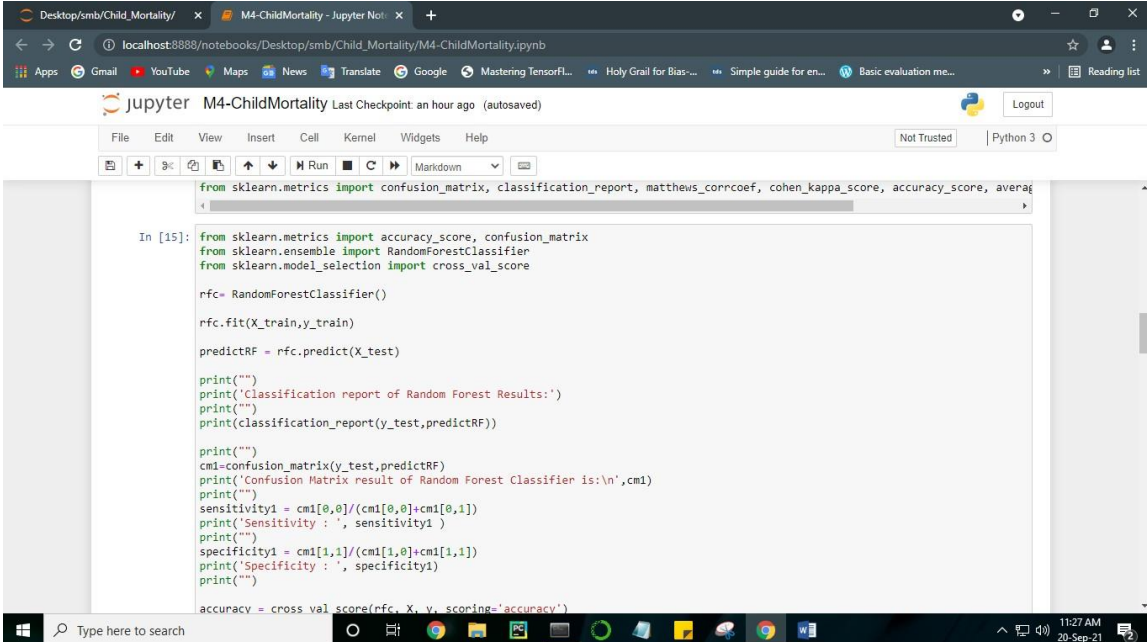
RANDOM FOREST CLASSIFIER

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set. Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm

combines multiple algorithm of the same type i.e. multiple decision trees, resulting in a forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.

The following are the basic steps involved in performing the random forest algorithm:

- ☐ Pick N random records from the dataset.
- ☐ Build a decision tree based on these N records.
- ☐ Choose the number of trees you want in your algorithm and repeat steps 1 and 2.
- ☐ In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.



```
from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef, cohen_kappa_score, accuracy_score, average_precision_score

In [15]: from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score

rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)

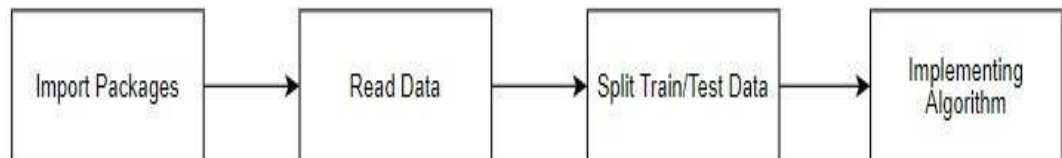
predictRF = rfc.predict(X_test)

print("")
print('Classification report of Random Forest Results:')
print("")
print(classification_report(y_test, predictRF))

print("")
cm1 = confusion_matrix(y_test, predictRF)
print('Confusion Matrix result of Random Forest Classifier is:\n', cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1)
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(rfc, X, y, scoring='accuracy')
```

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

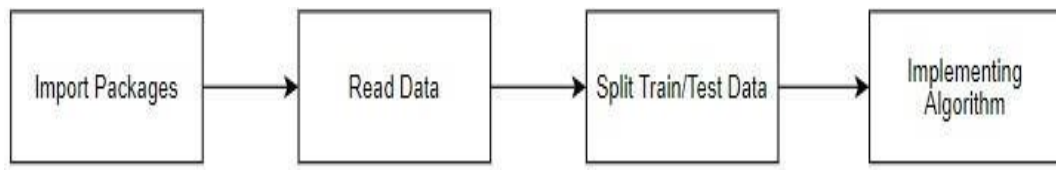
Naive Bayes algorithm:

- The Naive Bayes algorithm is an intuitive method that uses the probabilities of each attribute belonging to each class to make a prediction. It is the supervised learning approach you would come up with if you wanted to model a predictive modeling problem probabilistically.
- Naive bayes simplifies the calculation of probabilities by assuming that the probability of each attribute belonging to a given class

value is independent of all other attributes. This is a strong assumption but results in a fast and effective method.

- The probability of a class value given a value of an attribute is called the conditional probability. By multiplying the conditional probabilities together for each attribute for a given class value, we have a probability of a data instance belonging to that class. To make a prediction we can calculate probabilities of the instance belonging to each class and select the class value with the highest probability.
- Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets.
- Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location.
- Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered as naive. This assumption is called class conditional independence.

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

```
In [16]: from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import cross_val_score

gnb = GaussianNB()

gnb.fit(X_train,y_train)

predictNB = gnb.predict(X_test)

print("")
print('Classification report of Naive Bayes Results:')
print("")
print(classification_report(y_test,predictNB))

print("")
cm1=confusion_matrix(y_test,predictNB)
print('Confusion Matrix result of Naive Bayes is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1)
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(gnb, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Naive Bayes Algorithm is:",accuracy.mean() * 100)
NB=accuracy.mean() * 100
```

KNN algorithm

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand, but has a major drawback of becoming significantly slower as the size of the data in use grows.

The KNN algorithm can compete with the most accurate models because it makes highly accurate predictions. Therefore, you can use the KNN algorithm for applications that require high accuracy but that do not require a human-readable model. The quality of the predictions depends on the distance measure.

Most of the time, similar data points are close to each other. The KNN algorithm hinges on the assumption being true enough for the algorithm to be useful. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics.

The KNN Algorithm

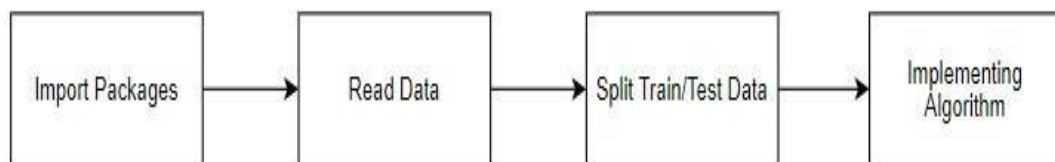
1. Load the data
2. Initialize K to your chosen number of neighbors
3. For each example in the data

Calculate the distance between the query example and the current example from the data.

Add the distance and the index of the example to an orderedcollection

4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
5. Pick the first K entries from the sorted collection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels

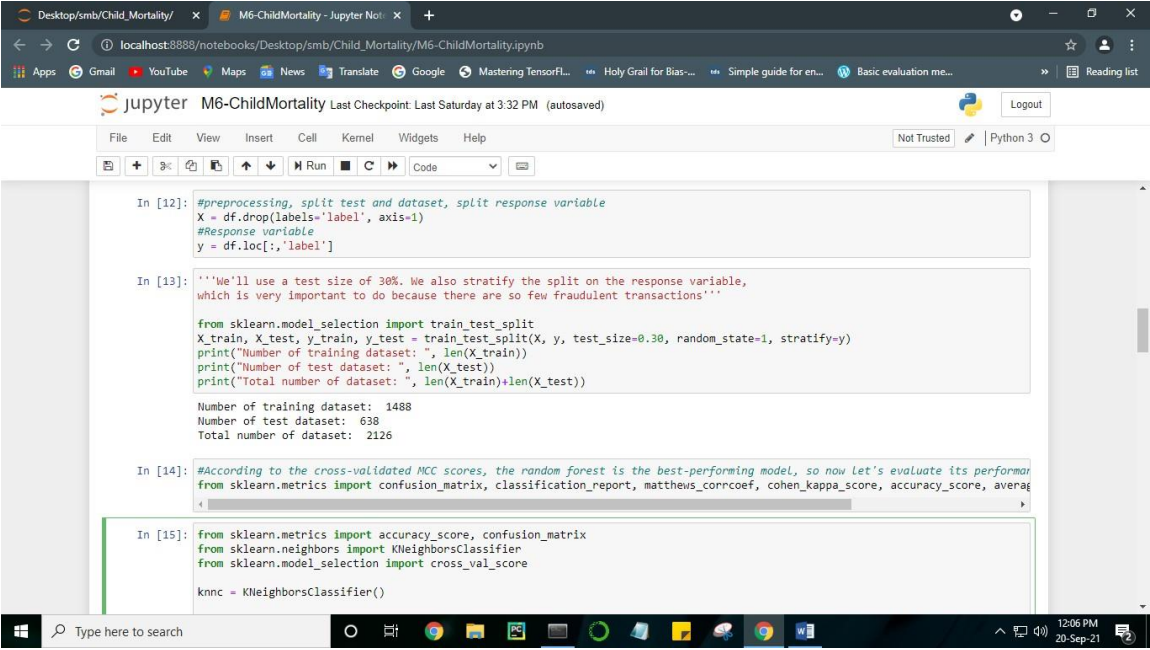
MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy



The screenshot shows a Jupyter Notebook titled "M6-ChildMortality" running on a local host. The notebook contains three code cells. The first cell (In [12]) performs data preprocessing by dropping the 'label' column from the dataset 'df' and splitting it into features 'X' and labels 'y'. The second cell (In [13]) uses `train_test_split` to create training and testing datasets with a 30% test size, stratified by the response variable. It prints the sizes of the training (1488), test (638), and total (2126) datasets. The third cell (In [14]) imports metrics for evaluation, including `confusion_matrix`, `classification_report`, `matthews_corrcoef`, `cohen_kappa_score`, `accuracy_score`, and `average_precision_score`. A fourth cell (In [15]) imports `accuracy_score` and `KNeighborsClassifier` from `sklearn.neighbors`, and `cross_val_score` from `sklearn.model_selection`. It also initializes a `KNeighborsClassifier` object named `knnc`.

```
In [12]: #preprocessing, split test and dataset, split response variable
X = df.drop(labels='label', axis=1)
#Response variable
y = df.loc[:, 'label']

In [13]: '''We'll use a test size of 30%. We also stratify the split on the response variable,
which is very important to do because there are so few fraudulent transactions'''

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=1, stratify=y)
print("Number of training dataset: ", len(X_train))
print("Number of test dataset: ", len(X_test))
print("Total number of dataset: ", len(X_train)+len(X_test))

Number of training dataset: 1488
Number of test dataset: 638
Total number of dataset: 2126

In [14]: #According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance
from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef, cohen_kappa_score, accuracy_score, average_precision_score

In [15]: from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score

knnc = KNeighborsClassifier()
```

CHAPTER-5

RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

Random Forest Algorithm

```

In [1]: #importing library Packages
import pandas as p
import numpy as n
import matplotlib.pyplot as plt
import seaborn as s

In [2]: import warnings
warnings.filterwarnings('ignore')

In [3]: data=p.read_csv('child.csv')

In [4]: data.head()

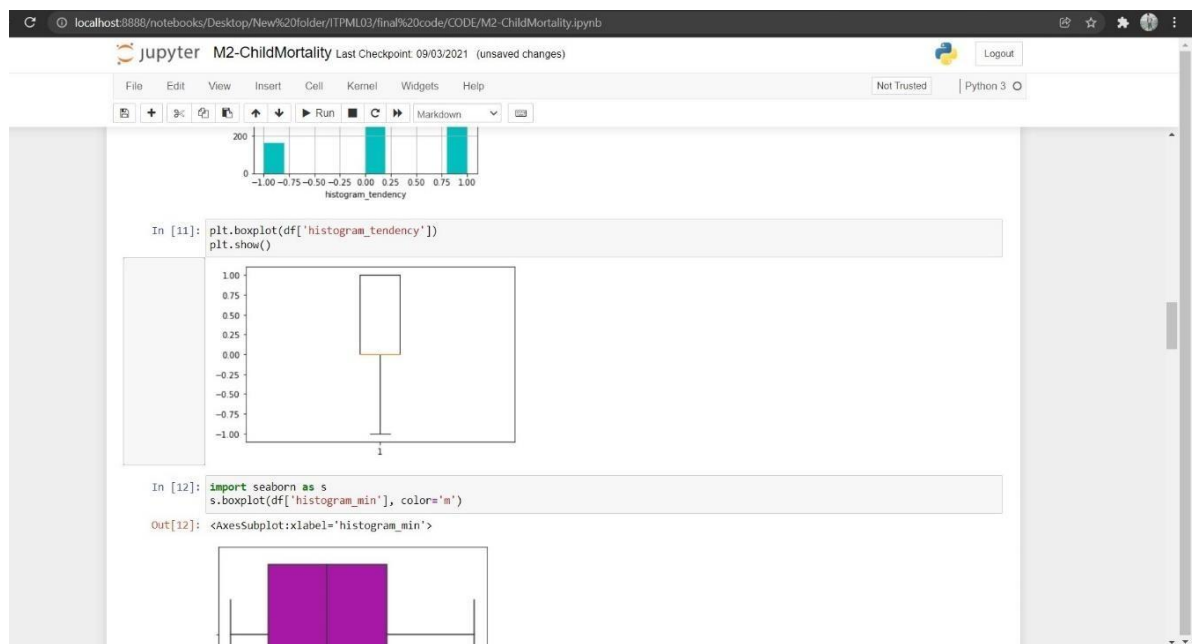
Out[4]:
  baseline_value  accelerations  fetal_movement  uterine_constrictions  light_decelerations  severe_decelerations  prolonged_decelerations  abnormal_short_term
0             120           0.000             0.0                   0.000             0.000             0.0                   0.0
1             132           0.006             0.0                   0.006             0.003             0.0                   0.0
2             133           0.003             0.0                   0.008             0.003             0.0                   0.0
3             134           0.003             0.0                   0.008             0.003             0.0                   0.0
4             132           0.007             0.0                   0.008             0.000             0.0                   0.0

5 rows x 22 columns

In [5]: data.shape
Out[5]: (2126, 22)

In [6]: df=data.dropna()
  
```

During the process at first we are including the libraries to the program , loading the data set by checking the data if there any null values it will be replaced by the mean and median values to that. we will see the data visually in the form of graph , charts e.t.c, . We will divide the data sets into two parts training and testing data sets in the ratio of 8:2 . we selected the algorithms according to the data sets divided . the algorithms are logistic regression, Random Forest, k-nearest neighbor, Naive bayes. By Using these algorithms we will train our data after training and testing data will be given to that model. If the model gives good accuracy That algorithm is given to the deployment. If the model does not give the better accuracy we will change to other algorithms. By the above process we can conclude that Random Forest algorithm given the best accuracy than the other three Algorithms.



localhost:8888/notebooks/Desktop/New%20folder/ITPML03/final%20code/CODE/M4-ChildMortality.ipynb

jupyter M4-ChildMortality Last Checkpoint: 09/04/2021 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3

```
print("")
accuracy = cross_val_score(rfc, X, y, scoring='accuracy')
print("Cross validation test results of accuracy:")
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Random Forest Classifier is:", accuracy.mean() * 100)
RF=accuracy.mean() * 100
```

Classification report of Random Forest Results:

	precision	recall	f1-score	support
0	0.97	0.99	0.98	497
1	0.97	0.89	0.93	141
accuracy			0.97	638
macro avg	0.97	0.94	0.96	638
weighted avg	0.97	0.97	0.97	638

Confusion Matrix result of Random Forest Classifier is:

```
[[493  4]
 [ 15 126]]
```

Sensitivity : 0.9919517102615694

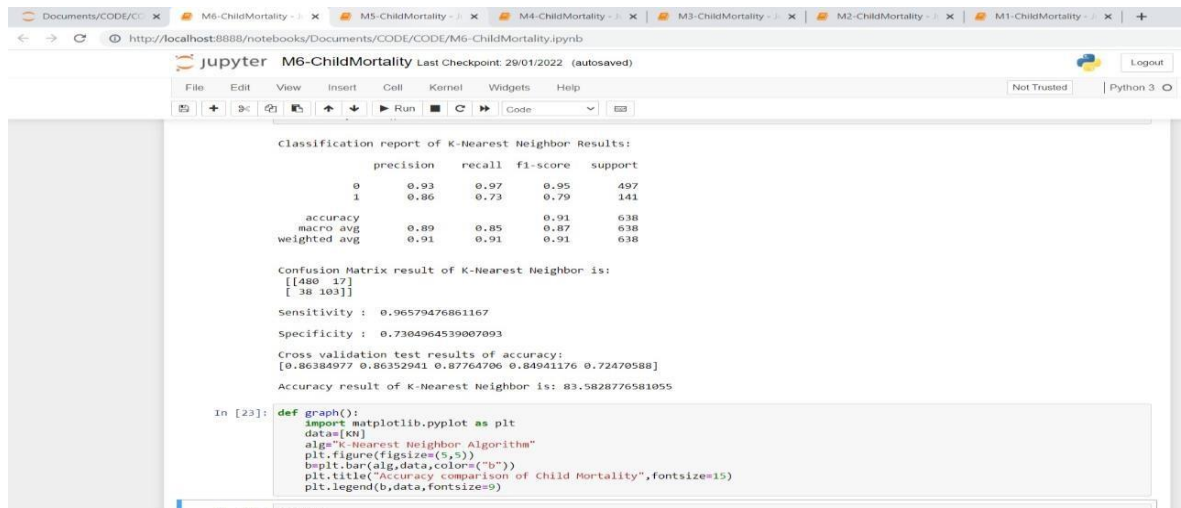
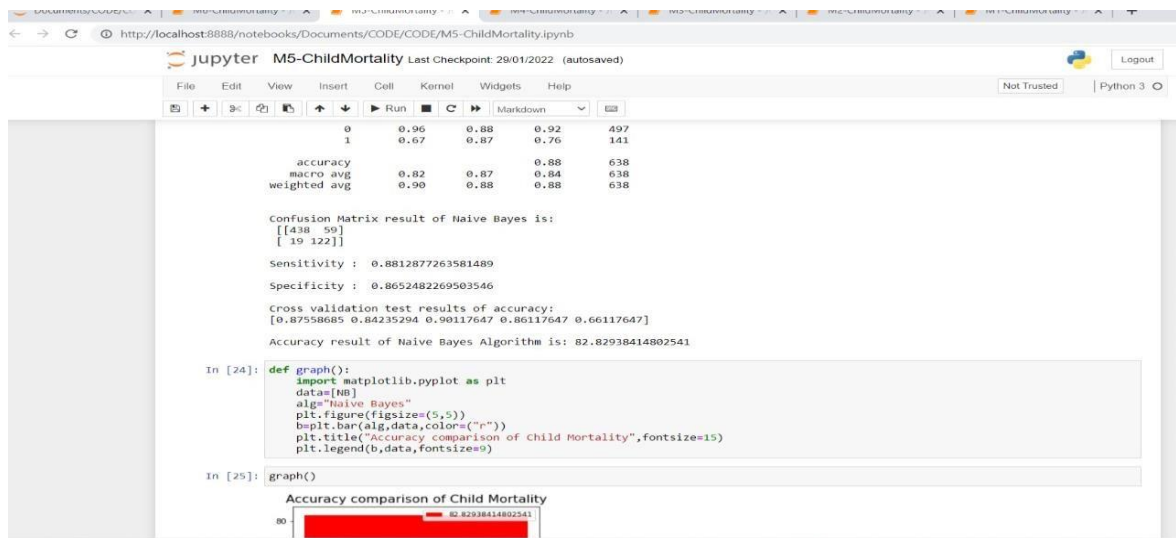
Specificity : 0.8936170212765957

Cross validation test results of accuracy:

```
[0.91314554 0.92235294 0.93882353 0.88      0.74823529]
```

Accuracy result of Random Forest Classifier is: 88.0511460922397

```
In [23]: def graph():
import matplotlib.pyplot as plt
```



PERFORMANCE ANALYSIS AND COMPARISON

LOGISTIC REGRESSION:

ACCURACY:85.3

RANDOM FOREST:

ACCURACY:88

NAIVE BAYES:

ACCURACY:82.82

K-NEAREST NEIGHBOUR:

ACCURACY:83.58

CHAPTER-6

SUMMARY AND CONCLUSIONS

Conclusion

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set is higher accuracy score is will be find out. This application can help to find the Prediction of Child Mortality.

Future Work

- ☐ Child Mortality prediction to connect with AI model.
- ☐ To automate this process by show the prediction result in web application or desktop application.
- ☐ To optimize the work to implement in Artificial Intelligence environment

REFERENCES

- [1]. Las Johansen B. Caluza, "Man-made brainpower Algorithm Application in Predicting Children Mortality: A Model Development", International Journal of Information Sciences and Application, ISSN 0974-2255 Volume 10, Number 1 (2018).
- [2]. Fikrewold H. Bitew, Samuel H. Nyarko, Lloyd Potter and Corey S. Streaks, "PC based insight approach for expecting under-five mortality determinants in Ethiopia: affirmation from the 2016 Ethiopian Demographic and Health Survey", Published: 04 November 2020.
- [3]. Cheyenne Mangold, Sarah Zoretic, Keerthi Thallapureddy, Axel Moreira, Kevin Chorath, Alvaro Moreira, "Mimicked insight Models for Predicting Neonatal Mortality: A Systematic Review".
- [4]. <https://doi.org/10.3389/fped.2021.644922>, Fu-Sheng Chou¹ and Laxmi V. Ghimire^{Machine}, "Learning for Mortality Prediction in Pediatric Myocarditis".
- [5]. Adeyinka Emmanuel Adegboosin, Bela Stantic, Jing Sun, "Sufficiency of significant learning techniques for expecting under-five mortality in 34 low-pay and focus pay countries".
- [6]. Chodziwadziwa Kabudula, Reesha Kara, Henry Wandera, Fidelia A. A. Dake, Justin Dansou, Dereje Danbe Debeko, Chipso Mufudza, "Appraisal of AI techniques for expecting the risk of child mortality in South Africa".
- [7]. <https://doi.org/10.1177/1550147720928897>, Muhammad Islam, Muhammad Usman, Muhammad Usman, Aaqif Afzaal Abbasi, Oh-Young Song, "Insightful assessment structure for exact appraisal of a

child passing rates for Internet of Things enabled wise clinical consideration systems”.

[8]. Rushikesh Gawande, Shreiya Indulkar, Himani Keswani, Malvika Khatri, Pallavi Saindane, “Appraisal and Prediction of Child Mortality in India”, e-ISSN: 2395-0056, p-ISSN: 2395-0072, Volume: 06 Issue: 03 | Mar 2019.

[9]. Dr. Smartson. P. NYONI, Thabani NYONI, “Assumption for Infant Mortality Rate in India Using a Machine Learning Technique”, ISSN (on the web): 2581-3048 Volume 5, Issue 3, pp 637-641, March-2021,<https://doi.org/10.47001/IRJIET/2021.503110>.

[10]. Dweepobotee Brahma, Debasri Mukherjee, “Early counsel signs: zeroing in on neonatal and child mortality using machine learning”,<https://doi.org/10.1080/00036846.2021.1958141>.

SOURCE CODE:

Module – 1

Pre-Processing

```
import pandas as p
```

```
import numpy as n
```

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

```
data=p.read_csv('child.csv')
```

```
data.head()
```

```
data.shape
```

```
data.isnull().sum()
```

Drop the Given Dataset

```
df=data.dropna()
```

`df.shape`

`df.head()`

`df.isnull().sum()`

`df.describe()`

`df.columns`

`df.baseline_value.unique()`

`df.accelerations.unique()`

`df.fetal_movement.unique()`

`df.uterine_contractions.unique()`

`df.light_decelerations.unique()`

`df.severe_decelerations.unique()`

```
df.prolongued_decelerations.unique()
```

```
df.abnormal_short_term_variability.unique()
```

```
df.mean_value_of_short_term_variability.unique()
```

```
df.percentage_of_time_with_abnormal_long_term_variability.  
unique()
```

```
df.mean_value_of_long_term_variability.unique()
```

```
df.histogram_width.unique()
```

```
df.histogram_min.unique()
```

```
df.histogram_max.unique()
```

```
df.histogram_number_of_peaks.unique()
```

```
df.histogram_number_of_zeroes.unique()
```

```
df.histogram_mode.unique()
```

```
df.histogram_mean.unique()
```

```
df.histogram_median.unique()
```

```
df.histogram_variance.unique()
```

```
df.histogram_tendency.unique()
```

```
df.fetal_health.unique()
```

```
df['label'] = df.fetal_health.map({1:0,2:1,3:1})
```

```
df.head()
```

```
del df['fetal_health']
```

```
df.head()
```

```
df.info()
```

```
df.duplicated()
```

```
sum(df.duplicated())
```

```
df.corr()
```

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

```
print("Minimum value of  
percentage_of_time_with_abnormal_long_term_variability  
is:",
```

```
df.percentage_of_time_with_abnormal_long_term_variability.  
min())
```

```
print("Maximum value of  
percentage_of_time_with_abnormal_long_term_variability  
is:",
```

```
df.percentage_of_time_with_abnormal_long_term_variability.  
max())
```

```
print("Minimum value of accelerations is:",  
df.accelerations.min())
```

```
print("Maximum value of accelerations is:",  
df.accelerations.max())
```

```
print("Minimum value of light_decelerations is:",  
df.light_decelerations.min())
```

```
print("Maximum value of light_decelerations is:",  
df.light_decelerations.max())
```

```
print("Minimum value of histogram_mean is:",
df.histogram_mean.min())

print("Maximum value of histogram_mean is:",
df.histogram_mean.max())
```

```
df.columns
```

```
from sklearn.preprocessing import LabelEncoder

var_mod = ['baseline_value', 'accelerations', 'fetal_movement',
'uterine_contractions', 'light_decelerations',
'severe_decelerations',
'prolongued_decelerations', 'abnormal_short_term_variability',
'mean_value_of_short_term_variability',
'percentage_of_time_with_abnormal_long_term_variability',
'mean_value_of_long_term_variability', 'histogram_width',
'histogram_min', 'histogram_max',
'histogram_number_of_peaks',
'histogram_number_of_zeroes', 'histogram_mode',
'histogram_mean',
'histogram_median', 'histogram_variance',
'histogram_tendency',
'label']

le = LabelEncoder()
```

```
for i in var_mod:  
    df[i] = le.fit_transform(df[i]).astype(int)
```

```
df.head(5)
```

Module – 2

Visualization

```
import pandas as p  
import numpy as n  
import seaborn as s  
import matplotlib.pyplot as plt  
import warnings  
warnings.filterwarnings('ignore')  
data=p.read_csv('child.csv')  
data.shape  
df=data.dropna  
p.crosstab(df.histogram_variance,df.histogram_tendency)  
#Histogram for histogram_max & histogram_min  
df['histogram_min'].hist(figsize=(5,5), color='m', alpha=1)  
plt.xlabel('histogram_min')  
plt.ylabel('histogram_max')  
plt.title('histogram_min & histogram_max')  
df['histogram_tendency'].hist(figsize=(5,5), color='c', alpha=1)
```

```

plt.xlabel('histogram_tendency')
plt.ylabel('histogram_variance')
plt.title('histogram_tendency & histogram_variance')
plt.boxplot(df['histogram_tendency'])
plt.show()

import seaborn as s
s.boxplot(df['histogram_min'], color='m')
df.columns

fig, ax = plt.subplots(figsize=(16,8))
ax.scatter(df['abnormal_short_term_variability'],df['histogram_
number_of_peaks'])
ax.set_xlabel('abnormal_short_term_variability')
ax.set_ylabel('histogram_number_of_peaks')
plt.show()

#Propagation by variable
def PropByVar(df, variable):
    dataframe_pie = df[variable].value_counts()
    ax = dataframe_pie.plot.pie(figsize=(10,10),
    autopct='%1.2f%%', fontsize = 12)
    ax.set_title(variable + '\n', fontsize = 15)
    return n.round(dataframe_pie/df.shape[0]*100,2)

PropByVar(df, 'uterine_contractions')

```

```

fig, ax = plt.subplots(figsize=(25,12))
s.heatmap(df.corr(), ax=ax, annot=True)

plt.plot(df["histogram_tendency"], df["histogram_variance"],
color='r')

plt.xlabel('histogram_tendency')
plt.ylabel('histogram_variance')
plt.title('Child Mortality')
plt.show()

df['label'] = df.fetal_health.map({1:0,2:1,3:1})
del df['fetal_health']

df.columns

#preprocessing, split test and dataset, split response variable
X = df.drop(labels='label', axis=1)

#Response variable
y = df.loc[:, 'label']

'''We'll use a test size of 30%. We also stratify the split on the
response variable,
which is very important to do because there are so few
fraudulent transactions'''

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.30, random_state=1, stratify=y)

print("Number of training dataset: ", len(X_train))

print("Number of test dataset: ", len(X_test))

```

```
print("Total number of dataset: ", len(X_train)+len(X_test))
```

Module – 3

Logistic Regression Algorithm

```
import pandas as p
```

```
import numpy as n
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as s
```

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

```
data=p.read_csv('child.csv')
```

```
df=data.dropna()
```

```
df.shape
```

```
df['label'] = df.fetal_health.map({1:0,2:1,3:1})
```

```
del df['fetal_health']
```

```
df.columns
```

```
#preprocessing, split test and dataset, split response variable
```

```
X = df.drop(labels='label', axis=1)
```

```
#Response variable
```

```
y = df.loc[:, 'label']
```

```
"""We'll use a test size of 20%. We also stratify the split on the  
response variable,
```

which is very important to do because there are so few fraudulent transactions"

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20, random_state=1, stratify=y)

print("Number of training dataset: ", len(X_train))

print("Number of test dataset: ", len(X_test))

print("Total number of dataset: ", len(X_train)+len(X_test))

#According to the cross-validated MCC scores, the random
forest is the best-performing model, so now let's evaluate its
performance on the test set.

from sklearn.metrics import confusion_matrix,
classification_report, matthews_corrcoef,
cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score

from sklearn.metrics import accuracy_score, confusion_matrix

from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import cross_val_score

logR= LogisticRegression()

logR.fit(X_train,y_train)

predictLR = logR.predict(X_test)

print("")

print('Classification report of Logistic Regression Results:')

print("")
```

```

print(classification_report(y_test,predictLR))

print("")

cm1=confusion_matrix(y_test,predictLR)

print('Confusion Matrix result of Logistic Regression is:\n',cm1)

print("")

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])

print('Sensitivity : ', sensitivity1 )

print("")

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])

print('Specificity : ', specificity1)

print("")

accuracy = cross_val_score(logR, X, y, scoring='accuracy')

print('Cross validation test results of accuracy:')

print(accuracy)

#get the mean of each fold

print("")

print("Accuracy      result      of      Logistic      Regression
is:",accuracy.mean() * 100)

LR=accuracy.mean() * 100

def graph():

import matplotlib.pyplot as plt

data=[LR]

alg="Logistic Regression"

```

```

plt.figure(figsize=(5,5))

b=plt.bar(alg,data,color=("y"))

plt.title("Accuracy comparison of Child Mortality",fontsize=15)

plt.legend(b,data,fontsize=9)

graph()

TP = cm1[0][0]

FP = cm1[1][0]

FN = cm1[1][1]

TN = cm1[0][1]

print("True Positive :",TP)

print("True Negative :",TN)

print("False Positive :",FP)

print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)

TNR = TN/(TN+FP)

FPR = FP/(FP+TN)

FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)

print("True Negative Rate :",TNR)

print("False Positive Rate :",FPR)

print("False Negative Rate :",FNR)

print("")

```

$PPV = TP/(TP+FP)$

$NPV = TN/(TN+FN)$

```
print("Positive Predictive Value :",PPV)
```

```
print("Negative predictive value :",NPV)
```

```
def plot_confusion_matrix(cm1, title='Confusion matrix-  
Logistic_Regression', cmap=plt.cm.Blues):
```

```
target_names=['Predict','Actual']
```

```
plt.imshow(cm1, interpolation='nearest', cmap=cmap)
```

```
plt.title(title)
```

```
plt.colorbar()
```

```
tick_marks = n.arange(len(target_names))
```

```
plt.xticks(tick_marks, target_names, rotation=45)
```

```
plt.yticks(tick_marks, target_names)
```

```
plt.tight_layout()
```

```
plt.ylabel('True label')
```

```
plt.xlabel('Predicted label')
```

```
cm1=confusion_matrix(y_test, predictLR)
```

```
print('Confusion matrix-Logistic_Regression:')
```

```
print(cm1)
```

```
plot_confusion_matrix(cm1)
```

Module – 4

Random Forest Classifier

```

#importing Library Packages

import pandas as p

import numpy as n

import matplotlib.pyplot as plt

import seaborn as s

import warnings

warnings.filterwarnings('ignore')

data=p.read_csv('child.csv')

df=data.dropna()

df.shape

df.fetal_health.unique()

df['label'] = df.fetal_health.map({1:0,2:1,3:1})

df.columns

del df['fetal_health']

#preprocessing, split test and dataset, split response variable

X = df.drop(labels='label', axis=1)

#Response variable

y = df.loc[:, 'label']

"""We'll use a test size of 30%. We also stratify the split on the
response variable,

which is very important to do because there are so few
fraudulent transactions"""

from sklearn.model_selection import train_test_split

```

```

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.30, random_state=1, stratify=y)

print("Number of training dataset: ", len(X_train))

print("Number of test dataset: ", len(X_test))

print("Total number of dataset: ", len(X_train)+len(X_test))

#According to the cross-validated MCC scores, the random
forest is the best-performing model, so now let's evaluate its
performance on the test set.

from sklearn.metrics import confusion_matrix,
classification_report, matthews_corrcoef,
cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score

from sklearn.metrics import accuracy_score, confusion_matrix

from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import cross_val_score

rfc= RandomForestClassifier()

rfc.fit(X_train,y_train)

predictRF = rfc.predict(X_test)

print("")

print('Classification report of Random Forest Results:')

print("")

print(classification_report(y_test,predictRF))

print("")

cm1=confusion_matrix(y_test,predictRF)

```

```

print('Confusion Matrix result of Random Forest Classifier
is:\n',cm1)

print("")

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])

print('Sensitivity : ', sensitivity1 )

print("")

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])

print('Specificity : ', specificity1)

print("")

accuracy = cross_val_score(rfc, X, y, scoring='accuracy')

print('Cross validation test results of accuracy:')

print(accuracy)

#get the mean of each fold

print("")

print("Accuracy result of Random Forest Classifier
is:",accuracy.mean() * 100)

RF=accuracy.mean() * 100

def graph():

import matplotlib.pyplot as plt

data=[RF]

alg="Random Forest Classifier"

plt.figure(figsize=(5,6))

b=plt.bar(alg,data,color=("m"))

```

```

plt.title("Accuracy comparison of Child Mortality",fontsize=15)

plt.legend(b,data,fontsize=9)

graph()

TP = cm1[0][0]
FP = cm1[1][0]
FN = cm1[1][1]
TN = cm1[0][1]

print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)
NPV = TN/(TN+FN)

```

```

print("Positive Predictive Value :",PPV)

print("Negative predictive value :",NPV)

def plot_confusion_matrix(cm1, title='Confusion matrix-
RandomForestClassifier', cmap=plt.cm.Blues):
    target_names=['Predict','Actual']

    plt.imshow(cm1, interpolation='nearest', cmap=cmap)

    plt.title(title)

    plt.colorbar()

    tick_marks = np.arange(len(target_names))

    plt.xticks(tick_marks, target_names, rotation=45)

    plt.yticks(tick_marks, target_names)

    plt.tight_layout()

    plt.ylabel('True label')

    plt.xlabel('Predicted label')

    cm1=confusion_matrix(y_test, predictRF)

    print('Confusion matrix-RandomForestClassifier:')

    print(cm1)

    plot_confusion_matrix(cm1)

```

Module – 5

Naive Bayes Algorithm

```
import pandas as p
```

```

import numpy as n

import matplotlib.pyplot as plt

import seaborn as s

import warnings

warnings.filterwarnings('ignore')

data= p.read_csv('child.csv')

df=data.dropna()

df.shape

df.fetal_health.unique()

p.Categorical(df['fetal_health']).describe()

df.fetal_health.value_counts()

df['label']=df.fetal_health.map({1:0, 2:1, 3:1})

del df['fetal_health']

df.columns

#preprocessing, split test and dataset, split response variable

X = df.drop(labels='label', axis=1)

#Response variable

y = df.loc[:, 'label']

"""We'll use a test size of 30%. We also stratify the split on the
response variable,

which is very important to do because there are so few
fraudulent transactions"""

from sklearn.model_selection import train_test_split

```

```

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.30, random_state=1, stratify=y)

print("Number of training dataset: ", len(X_train))

print("Number of test dataset: ", len(X_test))

print("Total number of dataset: ", len(X_train)+len(X_test))

#According to the cross-validated MCC scores, the random
forest is the best-performing model, so now let's evaluate its
performance on the test set.

from sklearn.metrics import confusion_matrix,
classification_report, matthews_corrcoef,
cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score

from sklearn.metrics import accuracy_score, confusion_matrix

from sklearn.naive_bayes import GaussianNB

from sklearn.model_selection import cross_val_score

gnb = GaussianNB()

gnb.fit(X_train,y_train)

predictNB = gnb.predict(X_test)

print("")

print('Classification report of Naive Bayes Results:')

print("")

print(classification_report(y_test,predictNB))

print("")

cm1=confusion_matrix(y_test,predictNB)

```

```

print('Confusion Matrix result of Naive Bayes is:\n',cm1)

print("")

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])

print('Sensitivity : ', sensitivity1 )

print("")

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])

print('Specificity : ', specificity1)

print("")

accuracy = cross_val_score(gnb, X, y, scoring='accuracy')

print('Cross validation test results of accuracy:')

print(accuracy)

#get the mean of each fold

print("")

print("Accuracy result of Naive Bayes Algorithm
is:",accuracy.mean() * 100)

NB=accuracy.mean() * 100

def graph():

import matplotlib.pyplot as plt

data=[NB]

alg="Naive Bayes"

plt.figure(figsize=(5,5))

b=plt.bar(alg,data,color=("r"))

plt.title("Accuracy comparison of Child Mortality",fontsize=15)

```

```

plt.legend(b,data,fontsize=9)

graph()

TP = cm1[0][0]
FP = cm1[1][0]
FN = cm1[1][1]
TN = cm1[0][1]

print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)
NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)

```

```

print("Negative predictive value :",NPV)

def plot_confusion_matrix(cm1, title='Confusion matrix-Naive
Bayes', cmap=plt.cm.Blues):
    target_names=['Predict','Actual']

    plt.imshow(cm1, interpolation='nearest', cmap=cmap)

    plt.title(title)

    plt.colorbar()

    tick_marks = n.arange(len(target_names))

    plt.xticks(tick_marks, target_names, rotation=45)

    plt.yticks(tick_marks, target_names)

    plt.tight_layout()

    plt.ylabel('True label')

    plt.xlabel('Predicted label')

cm1=confusion_matrix(y_test, predictNB)

print('Confusion matrix-Naive Bayes:')

print(cm1)

plot_confusion_matrix(cm1)

```

Module – 6

K – Nearest Neighbour Algorithm

```
import pandas as p
```

```
import numpy as n
```

```

import matplotlib.pyplot as plt

import seaborn as s

import warnings

warnings.filterwarnings('ignore')

data=p.read_csv('child.csv')

df=data.dropna()

df.fetal_health.unique()

p.Categorical(df['fetal_health']).describe()

df.fetal_health.value_counts()

df['label']=df.fetal_health.map({1:0, 2:1, 3:1})

del df['fetal_health']

df.columns

#preprocessing, split test and dataset, split response variable

X = df.drop(labels='label', axis=1)

#Response variable

y = df.loc[:, 'label']

'''We'll use a test size of 30%. We also stratify the split on the
response variable,

which is very important to do because there are so few
fraudulent transactions'''

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.30, random_state=1, stratify=y)

print("Number of training dataset: ", len(X_train))

```

```

print("Number of test dataset: ", len(X_test))

print("Total number of dataset: ", len(X_train)+len(X_test))

#According to the cross-validated MCC scores, the random
forest is the best-performing model, so now let's evaluate its
performance on the test set.

from sklearn.metrics import confusion_matrix,
classification_report, matthews_corrcoef,
cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score

from sklearn.metrics import accuracy_score, confusion_matrix

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import cross_val_score

knnc = KNeighborsClassifier()

knnc.fit(X_train,y_train)

predictK = knnc.predict(X_test)

print("")

print('Classification report of K-Nearest Neighbor Results:')

print("")

print(classification_report(y_test,predictK))

print("")

cm1=confusion_matrix(y_test,predictK)

print('Confusion Matrix result of K-Nearest Neighbor is:\n',cm1)

print("")

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])

```

```

print('Sensitivity : ', sensitivity1 )

print("")

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])

print('Specificity : ', specificity1)

print("")

accuracy = cross_val_score(knnc, X, y, scoring='accuracy')

print('Cross validation test results of accuracy:')

print(accuracy)

#get the mean of each fold

print("")

print('Accuracy      result      of      K-Nearest      Neighbor
is:',accuracy.mean()*100)

KN=accuracy.mean()*100

def graph():

import matplotlib.pyplot as plt

data=[KN]

alg="K-Nearest Neighbor Algorithm"

plt.figure(figsize=(5,5))

b=plt.bar(alg,data,color=("b"))

plt.title("Accuracy comparison of Child Mortality",fontsize=15)

plt.legend(b,data,fontsize=9)

graph()

TP = cm1[0][0]

```

```

FP = cm1[1][0]
FN = cm1[1][1]
TN = cm1[0][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV = TN/(TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)
def plot_confusion_matrix(cm1, title='Confusion matrix-K-Nearest Neighbor', cmap=plt.cm.Blues):

```

```

target_names=['Predict','Actual']

plt.imshow(cm1, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick_marks = n.arange(len(target_names))

plt.xticks(tick_marks, target_names, rotation=45)

plt.yticks(tick_marks, target_names)

plt.tight_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

cm1=confusion_matrix(y_test, predictK)

print('Confusion matrix-K-Nearest Neighbor:')

print(cm1)

plot_confusion_matrix(cm1)

```

HTML Code:

```

<!DOCTYPE html>

<html >

<!--From https://codepen.io/frytyler/pen/EGdtg-->

<head>

<meta charset="UTF-8">

```

```
<title>TITLE</title>
```

```
<link      rel="stylesheet"      href="{{      url_for('static',  
filename='css/bootstrap.min.css') }}">
```

```
<link href='https://fonts.googleapis.com/css?family=Pacifico'  
rel='stylesheet' type='text/css'>
```

```
<link href='https://fonts.googleapis.com/css?family=Arimo'  
rel='stylesheet' type='text/css'>
```

```
<link href='https://fonts.googleapis.com/css?family=Hind:300'  
rel='stylesheet' type='text/css'>
```

```
<link  
href='https://fonts.googleapis.com/css?family=Open+Sans+C  
ondensed:300' rel='stylesheet' type='text/css'>
```

```
<style>
```

```
.back{
```

```
background-image:      url("{{      url_for('static',  
filename='image/image.jpg') }}" );
```

```
background-repeat:no-repeat;
```

```
background-size:cover;
```

```
}
```

```
.white{
```

```
color:white;
```

```
}
```

```
.nspace{
```

```
margin:15px 15px 30px 30px;
```

```

padding:9px 10px;
background: palegreen;
width:500px
}

.space{
margin:10px 30px;
padding:10px 10px;
background: palegreen;
width:500px
}

.gap{
padding:10px 20px;
}

</style>

</head>

<body class="back">

<div>

<div class="jumbotron">

<h1 style="text-align:center">PREDICTION OF CHILD
MORTALITY </h1>

</div>

<!-- Main Input For Receiving Query to our ML -->

```

```

<form class="form-group" action="{{
url_for('predict')}}method="post">

<div class="row">

<div class="gap col-md-6 ">

<label class="white" for="">baseline_value</label>

<input type="number" class="space form-control" step="0.01"
name="baseline_value" placeholder="baseline_value"
required="required" /><br>

<label class="white" for="">accelerations</label>

<input type="number" class="space form-control" step="0.01"
name="accelerations" placeholder="accelerations"
required="required" /><br>

<label class="white" for="">fetal_movement</label>

<input type="number" class="space form-control" step="0.01"
name="fetal_movement" placeholder="fetal_movement"
required="required" /><br>

<label class="white" for="">uterine_contractions</label>

<input type="number" class="space form-control" step="0.01"
name="uterine_contractions"
placeholder="uterine_contractions" required="required"
/><br>

<label class="white" for="">light_decelerations</label>

<input type="number" class="space form-control" step="0.01"
name="light_decelerations" placeholder="light_decelerations"
required="required" /><br>

```

```

<label class="white" for="">severe_decelerations</label>

<input type="number" class="space form-control" step="0.01"
name="severe_decelerations"
placeholder="severe_decelerations"      required="required"
/><br>

<label                                class="white"
for="">prolongued_decelerations</label>

<input type="number" class="space form-control" step="0.01"
name="prolongued_decelerations"
placeholder="prolongued_decelerations" required="required"
/><br>

<label                                class="white"
for="">abnormal_short_term_variability</label>

<input type="number" class="space form-control" step="0.01"
name="abnormal_short_term_variability"
placeholder="abnormal_short_term_variability"
required="required" /><br>

<label                                class="white"
for="">mean_value_of_short_term_variability</label>

<input type="number" class="space form-control" step="0.01"
name="mean_value_of_short_term_variability"
placeholder="mean_value_of_short_term_variability"
required="required" /><br>

<label                                class="white"
for="">percentage_of_time_with_abnormal_long_term_variab
ility</label>

```

```

<input type="number" class="space form-control" step="0.01"
name="percentage_of_time_with_abnormal_long_term_varia
bility"
placeholder="percentage_of_time_with_abnormal_long_term
_variability" required="required" /><br>

<label                                     class="white"
for="">mean_value_of_long_term_variability</label>

<input type="number" class="space form-control" step="0.01"
name="mean_value_of_long_term_variability"
placeholder="mean_value_of_long_term_variability"
required="required" /><br>

</div>

<div class="gap col-md-6">

<label class="white" for="">histogram_width</label>

<input type="number" class="space form-control" step="0.01"
name="histogram_width"      placeholder="histogram_width"
required="required" /><br>

<label class="white" for="">histogram_min</label>

<input type="number" class="space form-control" step="0.01"
name="histogram_min"      placeholder="histogram_min"
required="required" /><br>

<label class="white" for="">histogram_max</label>

<input type="number" class="space form-control" step="0.01"
name="histogram_max"      placeholder="histogram_max"
required="required" /><br>

```

```

<label class="white"
for="">histogram_number_of_peaks</label>

<input type="number" class="space form-control" step="0.01"
name="histogram_number_of_peaks"
placeholder="histogram_number_of_peaks"
required="required" /><br>

<label class="white"
for="">histogram_number_of_zeroes</label>

<input type="number" class="space form-control" step="0.01"
name="histogram_number_of_zeroes"
placeholder="histogram_number_of_zeroes"
required="required" /><br>

<label class="white" for="">histogram_mode</label>

<input type="number" class="space form-control" step="0.01"
name="histogram_mode" placeholder="histogram_mode"
required="required" /><br>

<label class="white" for="">histogram_mean</label>

<input type="number" class="space form-control" step="0.01"
name="histogram_mean" placeholder="histogram_mean"
required="required" /><br>

<label class="white" for="">histogram_median</label>

<input type="number" class="space form-control" step="0.01"
name="histogram_median" placeholder="histogram_median"
required="required" /><br>

<label class="white" for="">histogram_variance</label>

```

```

<input type="number" class="space form-control" step="0.01"
name="histogram_variance"
placeholder="histogram_variance" required="required" /><br>
<label class="white" for="">histogram_tendency</label>
<input type="number" class="space form-control" step="0.01"
name="histogram_tendency"
placeholder="histogram_tendency"          required="required"
/><br>
</div>
</div>
<div style="padding:2% 35%">
<button type="submit" class="btn btn-success btn-block"
style="width:350px;padding:20px">Predict</button>
</div>
</form>
<br>
<br>
<div style="background:skyblue;padding:2% 40%">
{{ prediction_text }}
</div>
</div>
</body>
</html>

```

Flask Deploy:

```
import numpy as np

from flask import Flask, request, jsonify, render_template

import pickle

import joblib

app = Flask(__name__)

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

@app.route('/')

def home():

    return render_template('index.html')

@app.route('/predict',methods=['POST'])

def predict():

    """

    For rendering results on HTML GUI

    """

    int_features = [(x) for x in request.form.values()]

    final_features = [np.array(int_features)]

    print(final_features)

    prediction = model.predict(final_features)

    output = prediction[0]

    if output==1:

        output='Pathalogical'
```

else:

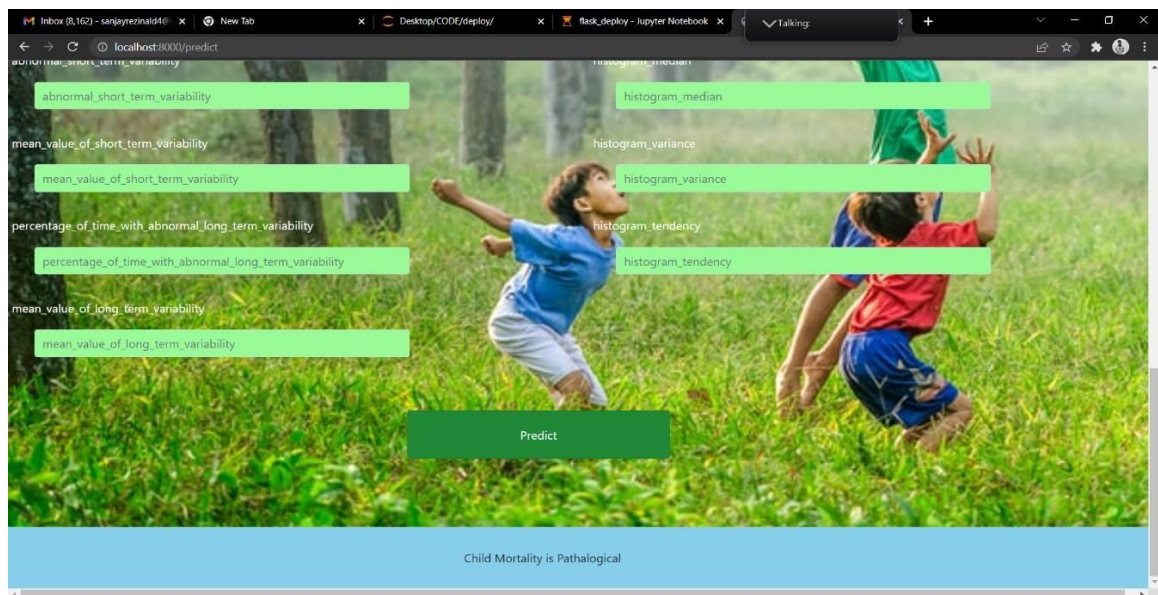
output="Normal"

return render_template('index.html', prediction_text=' Child
Mortality is {}'.format(output))

if __name__ == "__main__":

app.run(host="localhost", port=7000)

SCREENSHOTS





percentage_of_time_with_abnormal_long_term_variability

percentage_of_time_with_abnormal_long_term_varial

mean_value_of_long_term_variability

mean_value_of_long_term_variability

histogram_tendency

histogram_tendency

Predict

Child Mortality is Normal

PLAGIARISM REPORT

CHILD MORTALITY PREDICTION USING MACHINE LEARNING

ORIGINALITY REPORT

4%

SIMILARITY INDEX

3%

INTERNET SOURCES

2%

PUBLICATIONS

0%

STUDENT PAPERS

PRIMARY SOURCES

1

[rxivist.org](https://www.rxiv.org/)

Internet Source

1%

2

Carlos Eduardo Beluzo, Luciana Correia Alves, Everton Silva, Rodrigo Campos Bresan et al. "Machine Learning to Predict Neonatal Mortality Using Public Health Data from São Paulo - Brazil", Cold Spring Harbor Laboratory, 2020

Publication

1%

3

www.ijres.org

Internet Source

1%

4

irjetblog.wordpress.com

Internet Source

1%

Exclude quotes Off

Exclude bibliography On

Exclude matches Off