

SURVIVAL PREDICTION OF A PERSON WITH COVID-19

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

by

BOMMARAJU VARSHITHA
(Reg. No. 37110106)

BONTHU NAVEENA
(Reg. No. 37110109)



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 I 12B Status by UGC I
Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600119.

MARCH 2021



**SATHYABAMA INSTITUTE OF SCIENCE
AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with "A" grade by NAAC | 12B Status by
UGC | Approved by AICTE

Jeppiaar Nagar, Rajiv Gandhi Salai,
Chennai - 600119

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 **Ms. BOMMARAJU VARSHITHA (Reg. No. 37110106)**, **Ms. BONTHU NAVEENA (Reg.No.37110109)** who carried out the project entitled "**SURVIVAL PREDICTION OF A PERSON WITH COVID-19**" under our supervision from November 2020 to March 2021.

Internal Guide

M. S. Roobini, M. E;(Ph.D.)

Head of the Department

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I, **BOMMARAJU VARSHITHA (Reg.No:37110106), BONTHU NAVEENA (Reg.No.37110109)** hereby declare that the Project Report entitled “**SURVIVAL PREDICTION OF A PERSON WITH COVID-19**” is done by me under the guidance of Ms. **M.S. Roobini, M. E;(Ph.D.)** and Dr. **Ajitha Dept of CSE at Sathyabama Institute of Science and Technology** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA** 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. L. Lakshmanan M.E., Ph.D.**, and **Dr. S. Vigneshwari 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 **Ms. M. S. Roobini, M. E;(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.

SURVIVAL PREDICTION OF A PERSON WITH COVID-19

ABSTRACT

The spread of COVID-19 and similar viruses poses new challenges for our society. There is a strong incentive towards safety measures that help to mitigate the outbreaks. The impact of COVID-19 pandemic has led scientists to produce a vast quantity of research aimed at understanding, monitoring, and containing the disease. Technology advancements have a rapid effect on every field of life, be it medical field or any other field. People all over the world are vulnerable to its consequences in future. It is imperative to develop a control system that will detect the coronavirus. One of the solutions to control the current havoc can be the diagnosis of disease with the help of various tools. Our model aims at transfer learning, model integration and classify the chance of person's survival taking the count of various factors and parameters including death and recovery rates of a person into consideration. The application of machine learning in the field of medical diagnosis is increasing gradually. This can be contributed primarily to the improvement in the classification and recognition systems used in disease diagnosis which is able to provide data that aids medical experts in early detection of fatal diseases and therefore, increase the survival rate of patients significantly. We would like to use recent algorithmic enhancements that were developed to predict the death and recovery rates and also implement a method of predicting the chances of survival of a person infected with COVID-19 which is alarming and also widely spread in current case scenario.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	i. ABSTRACT	v
	ii. LIST OF FIGURES	ix
1.	INTRODUCTION	1
	1.1 SURVIVAL PREDICTION	1
	1.2 EXISTING SYSTEMS	1
	1.2.1 CANCER	1
	1.2.2 HEART FAILURE	2
	1.2.3 BRAIN TUMOUR	3
	1.3 TYPES OF SURVIVAL MODELS	4
	1.3.1 STATISTICAL METHODS	4
	1.3.1.1 NON-PARAMETRIC	4
	1.3.1.2 SEMI- PARAMETRIC	5
	1.3.1.3 PARAMETRIC	5
2.	LITERATURE SURVEY	7
3.	AIM AND SCOPE OF PRESENT INVESTIGATION	10
	3.1 OVERVIEW	10
	3.2 OBJECTIVE	10
	3.3 REQUIREMENT ANALYSIS	11
	3.3.1 FUCTIONAL REQUIREMENT	11
	3.3.2 NON-FUNCTIONAL REQUIREMENT	12
	3.4 FEASIBLE STUDY	13

	3.4.1 ECONOMIC FEASIBILITY	13
	3.4.2 TECHNICAL FEASIBILITY	13
	3.4.3 SOCIAL FEASIBILITY	14
	3.4.4 LEGAL FEASIBILITY	14
	3.4.5 OPERATIONAL FEASIBILITY	14
4.	EXPERIMENTAL MATERIALS, METHODS AND ALGORITHMS USED	15
	4.1 SYSTEM ARCHITECTURE	15
	4.1.1 MODULE DESCRIPTION	15
	4.2 ALGORITHM AND IMPLEMENTATION	16
	4.2.1 COX REGRESSION	16
	4.2.2 HAZARD AND BASELINE HAZARD	16
	4.3 LIFELINE'S MODULE	17
	4.4 SOFTWARE AND HARDWARE REQUIREMENTS	17
	4.4.1 HARDWARE REQUIREMENT	17
	4.4.2 SOFTWARE REQUIREMENT	17
5.	RESULTS AND DISCUSSIONS, PERFORMANCE ANALYSIS	18
	5.1 RESULTS AND DISCUSSIONS	18
	5.2 PERFORMANCE ANALYSIS	18
	5.2.1 TEST PLAN	18
	5.2.2 INPUT DESIGN	18
	5.2.3 OUTPUT DESIGN	19
6.	CONCLUSION AND FUTURE WORK	20
	6.1 CONCLUSION	20

6.2 FUTURE WORK	20
REFERENCES	21
APPENDIX	23
i. SOURCE CODE	23
ii. SCREEN SHOTS	29

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
1.1	TAXONOMY OF ANALYSIS MODELS	6
3.1	COVID-19	10
3.2	FEASIBILITY STUDY TYPES	14
4.1	SYSTEM ARCHITECTURE	15
B.1	PYTHON CODE	29
B.2	COX REGRESSION APPLICATION	29
B.3	APPLICATION OF EPOCH TECHNIQUE	30
B.4	EPOCH TECHNIQUE- 20 ITERATIONS	30
B.5	SURVIVAL PREDICTION RATES	31
B.6	RMSD CURVES	31

CHAPTER 1

INTRODUCTION

1.1 SURVIVAL PREDICTION

The objective in survival analysis also referred to as reliability analysis in engineering is to establish a connection between covariates and the time of an event. The name survival analysis originates from clinical research, where predicting the time to death, i.e., survival, is often the main objective. Survival analysis is a type of regression problem (one wants to predict a continuous value), but with a twist. It differs from traditional regression by the fact that parts of the training data can only be partially observed – they are censored. As a highly contagious respiratory disease, COVID-19 has yielded high mortality rates since its emergence in December 2019. As the number of COVID-19 cases soars in epicenters, health officials are warning about the possibility of the designated treatment centers being overwhelmed by coronavirus patients. In this study, several computational techniques are implemented to analyze the survival characteristics of patients. The computational results agree with the outcome reported in early clinical reports released for a group of patients. The discharge-time prediction of COVID-19 patients was also evaluated using different machine-learning and statistical analysis methods. The results indicate that the Gradient Boosting survival model outperforms other models for patient survival prediction in this study. This project is aimed to help health officials make more educated decisions during the outbreak.

1.2 SOME OF THE EXISTING SYSTEMS ON SURVIVAL ANALYSIS

1.2.1 CANCER

Machine learning is a branch of artificial intelligence that employs a variety of statistical, probabilistic and optimization techniques that allows computers to “learn” from past examples and to detect hard-to-discern patterns from large, noisy or complex data sets. This capability is particularly well-suited to medical applications, especially those that depend on complex proteomic and genomic measurements. As a result, machine learning is frequently used in cancer diagnosis and detection. More recently machine learning has been applied to cancer prognosis and

prediction. This approach is particularly interesting as it is part of a growing trend towards personalized, predictive medicine. In assembling this review, a broad survey of the different types of machine learning methods being used, the types of data being integrated and the performance of these methods in cancer prediction and prognosis are conducted. A number of trends are noted, including a growing dependence on protein biomarkers and microarray data, a strong bias towards applications in prostate and breast cancer, and a heavy reliance on “older” technologies such artificial neural networks (ANNs) instead of more recently developed or more easily interpretable machine learning methods. A number of published studies also appear to lack an appropriate level of validation or testing. Among the better designed and validated studies, it is clear that machine learning methods can be used to substantially (15–25%) improve the accuracy of predicting cancer susceptibility, recurrence and mortality. At a more fundamental level, it is also evident that machine learning is also helping to improve our basic understanding of cancer development and progression.

1.2.2 HEART FAILURE

Heart failure (HF) is a frequent health problem with high morbidity and mortality, increasing prevalence and escalating healthcare costs. By calculating a HF survival risk score based on patient-specific characteristics from Electronic Health Records (EHRs), we can identify high-risk patients and apply individualized treatment and healthy living choices to potentially reduce their mortality risk. The Seattle Heart Failure Model (SHFM) is one of the most popular models to calculate HF survival risk that uses multiple clinical variables to predict HF prognosis and also incorporates impact of HF therapy on patient outcomes. Although the SHFM has been validated across multiple cohorts, these studies were primarily done using clinical trials databases that do not reflect routine clinical care in the community. Further, the impact of contemporary therapeutic interventions, such as beta-blockers or defibrillators, was incorporated in SHFM by extrapolation from external trials. In this study, we assess the performance of SHFM using EHRs at Mayo Clinic, and sought to develop a risk prediction model using machine learning techniques that applies routine clinical care data. Results show the models which were built

using EHR data are more accurate (11% improvement in AUC) with the convenience of being more readily applicable in routine clinical care. Furthermore, we demonstrate that new predictive markers (such as co-morbidities) when incorporated into our models improve prognostic performance significantly (8% improvement in AUC).

1.2.3 BRAIN TUMOUR

Gliomas are the most common primary brain malignancies, with different degrees of aggressiveness, variable prognosis and various heterogeneous histologic sub-regions, i.e., peritumoral edematous/invaded tissue, necrotic core, active and non-enhancing core. This intrinsic heterogeneity is also portrayed in their radio-phenotype, as their sub-regions are depicted by varying intensity profiles disseminated across multi-parametric magnetic resonance imaging (mpMRI) scans, reflecting varying biological properties. Their heterogeneous shape, extent, and location are some of the factors that make these tumors difficult to resect, and in some cases inoperable. The amount of resected tumor is a factor also considered in longitudinal scans, when evaluating the apparent tumor for potential diagnosis of progression. Furthermore, there is mounting evidence that accurate segmentation of the various tumor sub-regions can offer the basis for quantitative image analysis towards prediction of patient overall survival. This study assesses the state-of-the-art machine learning (ML) methods used for brain tumor image analysis in mpMRI scans, during the last seven instances of the International Brain Tumor Segmentation (BraTS) challenge, i.e., 2012-2018. Specifically, we focus on i) evaluating segmentations of the various glioma sub-regions in pre-operative mpMRI scans, ii) assessing potential tumor progression by virtue of longitudinal growth of tumor sub-regions, beyond use of the RECIST/RANO criteria, and iii) predicting the overall survival from pre-operative mpMRI scans of patients that underwent gross total resection. Finally, we investigate the challenge of identifying the best ML algorithms for each of these tasks, considering that apart from being diverse on each instance of the challenge, the multi-institutional mpMRI BraTS dataset has also been a continuously evolving/growing dataset.

1.3 TYPES OF SURVIVAL ANALYSIS MODELS

1.3.1 STATISTICAL METHODS

Survival analysis arises in many applied fields such as medicine, biology, engineering, epidemiology and economics. Survival time is defined as a time to event of interest such as death, relapse of disease, unemployment and completion of a task. The main characteristic of survival time is that it is censored due to the end of study or withdrawal during the period of study because we cannot follow-up the exact survival time for those who are still alive at the end of the study, or who are lost to follow-up during the period of study. However, it is known that the survival time of censored individuals is at least longer than the censoring time. There are many different types of censoring such as right-, left-, and interval-censoring. The most popular censoring mechanisms are right censoring, in which the lower limit of the exact survival time is observed, while the upper limit is observed for left censoring, and both lower and upper limits are used for interval-censoring. Many statistical methods have been developed for estimating survival functions, comparing survival curves between two groups, and modelling the survival data by regression, for association with risk factors, such as demographic and clinical predictors.

1.3.1.1 NON - PARAMETRIC

In survival analysis, nonparametric statistical inference is more extensively used to estimate the survival function, and compare survival curves between two or more groups. For example, both the Kaplan-Meier (KM) estimator, for a survivor function, and a log-rank test, for comparison of survivor functions, are derived by a nonparametric approach. However, if the appropriate distribution for survival data is assumed or pre-specified, the parametric approach is more appropriate. Non-parametric methods are used to analyze data when the distributional assumptions of more common procedures are not satisfied. For example, many statistical procedures assume that the underlying error distribution is Gaussian, hence the widespread use of means and standard deviations. It is more efficient when no suitable theoretical distributions known. Difficult to interpret; yields inaccurate estimates. Specific methods include Kaplan-Meier, Nelson-Aalen, Life-Table.

1.3.1.2 SEMI - PARAMETRIC

When the association of survival time with various risk factors is the main interest, the most popular model is a Cox regression, based on a semiparametric approach, since the effect of predictors on the hazard rate is parametrically specified, while the baseline hazard function is unspecified. The alternative fork estimates the hazard function from the data. This approach is referred to as a semi-parametric approach because while the hazard function is estimated non-parametrically, the functional form of the covariates is parametric. A variety of parametric approaches are also available under the assumed survival distributions, such as an accelerated failure time (AFT) model. Overall, all survival analysis approaches should take into account a censoring mechanism, when a statistical inference is made. The knowledge of the underlying distribution of survival times is not required. The semi-parametric model relies on some very clever partial likelihood calculations by Sir David Cox in 1972 and the method is often called Cox regression in his honor. It is also often referred to as proportional hazards regression to highlight a major assumption of this model. Cox regression is a much more popular choice than parametric regression, because the nonparametric estimate of the hazard function offers you much greater flexibility than most parametric approaches. The distribution of the outcome is unknown; not easy to interpret. Specific methods include Cox model, Regularized Cox, CoxBoost, Time Dependent Cox.

1.3.1.3 PARAMETRIC

A parametric model will provide somewhat greater efficiency, because you are estimating fewer parameters. It also provides you with the ability to extrapolate beyond the range of the data. Finally, if the parametric model matches some underlying mechanism associated with your data, you end up with more relevant interpretations of your model. Parametric methods have been used extensively to analyze survival data. The parametric distribution that has been most widely used in fitting survival data is the exponential. However, the exponential survival distribution does not generally provide a satisfactory fit to survival data. One reason is that the hazard rate in survival studies is not usually constant with respect to time.

In roughly the last decade, other survival distributions have been used to fit to survival data. They include the Weibull, gamma, lognormal and extreme value distributions. These distributions allow more flexibility than the exponential because their associated hazard rates are not constant with respect to time. Methods are presented for estimating the parameters of the survival distributions mentioned above in the cases for which the data are uncensored and censored. Furthermore, survival may be correlated with some other variable or variables such as age, weight, or a diagnostic characteristic such as white blood count. To account for the relationship between survival and the variables related to survival, survival models have been developed which take into consideration these concomitant variables. Easy to interpret, more efficient and accurate when the survival times follow a particular distribution. When the distribution assumption is violated, it may be inconsistent and can give sub-optimal results. Specific methods include Tobit, Buckley-James, Penalized regression, Accelerated Failure Time.

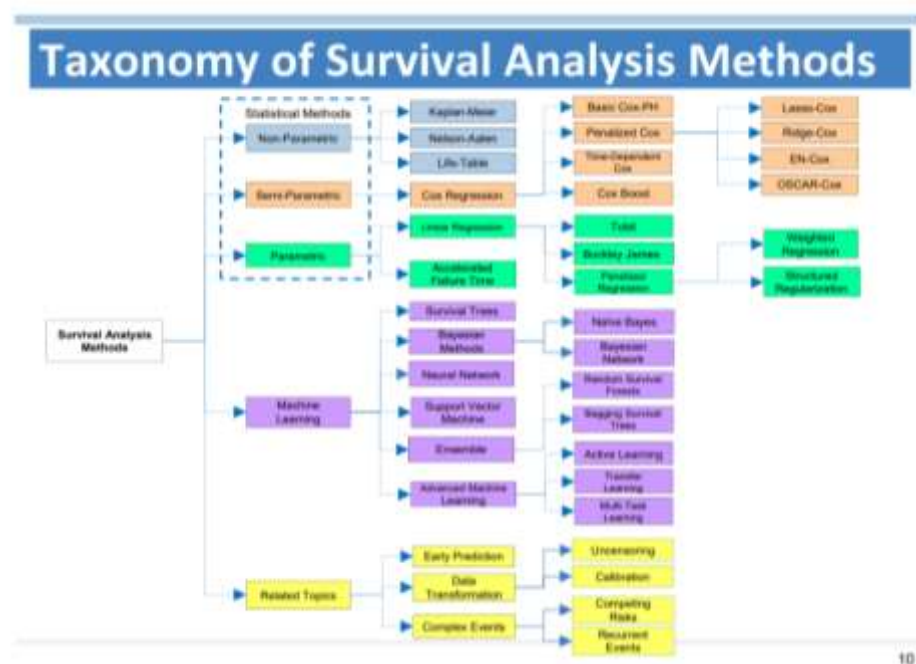


Fig. No. 1.1: Taxonomy of analysis methods

CHAPTER 2

LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool, it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then the next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system, the above consideration is taken into account for developing the proposed system. The major part of the project development sector considers and fully survey all the required needs for developing the project. For every project Literature survey is the most important sector in software development process. Before developing the tools and the associated designing it is necessary to determine and survey the time factor, resource requirement, man power, economy, and company strength. Once these things are satisfied and fully surveyed, then the next step is to determine about the software specifications in the respective system such as what type of operating system the project would require, and what are all the necessary software are needed to proceed with the next step such as developing the tools, and the associated operations.

[1] Vinay Chamolai, Vikas hasijja, vatsal gupta and Mohsen Guizani³

A thorough survey of the COVID-19 pandemic and the part of IoT, drones, man-made brainpower, blockchain and 5G administration. The World Health Organization (WHO) named the exceptional flare-up of the new Covid in 2019 and named it COVID-19, which puts numerous administrations around the planet in danger. Before, just Chinese nationals noticed the effect of the COVID-19 episode. Presently it involves extraordinary worry to pretty much every country on the planet.

[2] I. Douglas Moon, Sohail S. Chaudhry

Dissect network location issues and distance issues. As a matter of first importance, using different motivating representations, the test of accomplishment is viewed as the restriction of reality. This article utilizes another arrangement framework to present a progression of distance-based issues in a lucid way. This incorporates a few new issues. Second, investigate existing arrangements, which are simply appropriate to a portion of these issues. Finally, it is to shed some light on yet unstudied problems by exploring possible extensions of some of the known solution techniques or discussing varying degrees of difficulties involved.

[3] David Sayah, Stefan Irnich

This paper proposes a new compact formula for a different problem of distributing p . In addition, we have discussed two simple enhancements for the new formula: the simple limit of the correct distance can be used to reduce the size, and the model density can increase with the additional calculation time below. In addition, the new construction can be further strengthened by adding a valid inequality.

[4] Claudio Contardo

Minimize mixing to prove the effectiveness of the p -dispersion problem. Given the n points, the corresponding different matrix D size $n \times n$ and the number $p \geq 2$, the p -dispersion (pDP) problem involves selecting a set of points directly from p to the minimum between any pair. and very little. When p is the input question, the pDP is very difficult. We propose a minimally invasive approach to minimizing the problem in a series of small pDP solutions until better validation is achieved. The k -means algorithm is used to build and complete sets according to the algorithm. For small p -values, problems that can be solved by the proposed method are orders of magnitude larger than the latest pDP solution limit.

[5] Janelle S. Ayres

Survival of COVID-19, a concept of disease tolerance. An unexplained pneumonia epidemic broke out in Wuhan, China. In early January 2020, the virus was tracked and identified as a new type of coronavirus called SARS-CoV-2, the causative agent of COVID-19. Successfully responding to any outbreak of infectious disease requires a multidisciplinary approach. COVID-19 has put millions of dollars on the brink, and everyone is suddenly in an emergency.

[6] Ching-Fan Chung, Peter Sehmidt and Ann D. Witte 2

Review of some of the statistical methods involved in analyzing the length of time until a specified event occurs. We can illustrate the use of these techniques by modeling the length of time from an individual's release from prison until his recidivism. The statistical problems in this application are basically the same as those involved in a variety of other applications which focus on the time until an event occurs, such as the survival time of medical patients or the length of time until failure of equipment. We therefore use the simple phrase "survival time" to describe the type of variable of interest. The terms "failure time" and "waiting time" are also used in the statistical literature to describe the time until an event occurs, and we will use these terms interchangeably.

CHAPTER 3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

3.1 OVERVIEW

The main objective of this research is to design and implement an efficient algorithm in an effective way which is capable of predicting the survival rate of a person with affected with covid-19 based on their respective parameters like covid positive or negative, risks of heart disease, lung infections, diabetic, and many more. This can be monitored through a comfortable python shell interface. The proposed system has a great flexibility and ease using Cox- Regression to detect death rate and survival rate predictions and applying it to predict the final output detecting the survival of an individual. In medical field, manufacturing rates have risen day by day, and eventually diseases are also growing strongly. Detecting damaged components helps to create a software which can help patients to get more reliable results and it can exactly block the illness. It therefore enables doctors with modern disease detection techniques and contribute great returns as well. Therefore, this project mainly focuses on a healthy predicting and treatment outgrowth.



Fig. No. 3.1: COVID-19

3.2 OBJECTIVE

In the proposed system, we have mainly used an algorithm named cox-regression to detect the survival rate. This is done mainly by the process of taking death and recovered data as input and estimating the final output of the survival respectively.

In this project we use censoring which is most common in survival analysis. We can know the time when it occurred. It is a form of instance-based learning where the function is only locally approximated and all the computation is postponed until prediction.

3.3 REQUIREMENT ANALYSIS

Requirement analysis, also called requirement engineering, is the process of determining user expectations for a new modified product. It encompasses the tasks that determine the need for analyzing, documenting, validating and managing software or system requirements. The requirements should be documentable, actionable, measurable, testable and traceable related to identified business needs or opportunities and define to a level of detail, sufficient for system design.

3.3.1 FUNCTIONAL REQUIREMENT

It is a technical specification requirement for the software products. It is the first step in the requirement analysis process which lists the requirements of particular software systems including functional, performance and security requirements. The function of the system depends mainly on the quality hardware used to run the software with given functionality.

- **Usability**

It specifies how easy the system must be use. It is easy to ask queries in any format which is short or long, porter stemming algorithm stimulates the desired response for user.

- **Robustness**

It refers to a program that performs well not only under ordinary conditions but also under unusual conditions. It is the ability of the user to cope with errors for irrelevant queries during execution.

- **Security**

The state of providing protected access to resource is security. The system provides good security and unauthorized users cannot access the system there by providing high security.

➤ **Reliability**

It is the probability of how often the software fails. The measurement is often expressed in MTBF (Mean Time Between Failures). The requirement is needed in order to ensure that the processes work correctly and completely without being aborted. It can handle any load and survive and survive and even capable of working around any failure.

➤ **Compatibility**

It is supported by version above all web browsers. Using any web servers like localhost makes the system real-time experience.

➤ **Flexibility**

The flexibility of the project is provided in such a way that it has the ability to run on different environments being executed by different users.

➤ **Safety**

Safety is a measure taken to prevent trouble. Every query is processed in a secured manner without letting others to know one's personal information.

3.3.2 NON- FUNCTIONAL REQUIREMENT

➤ **Portability**

It is the usability of the same software in different environments. The project can be run in any operating system.

➤ **Performance**

These requirements determine the resources required, time interval, throughput and everything that deals with the performance of the system.

➤ **Accuracy**

The result of the requesting query is very accurate and high speed of retrieving information. The degree of security provided by the system is high and effective.

➤ **Maintainability**

Project is simple as further updates can be easily done without affecting its stability. Maintainability basically defines that how easy it is to maintain the system. It means that how easy it is to maintain the system, analyse, change

and test the application. Maintainability of this project is simple as further updates can be easily done without affecting its stability.

3.4 FEASIBLE STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. The feasibility study investigates the problem and the information needs of the stakeholders. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution. The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components.

3.4.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must justify. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The

developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.4.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

3.4.4 LEGAL FEASIBILITY

A feasibility study is an analysis that takes all of a project's relevant factors into account—including economic, technical, legal, and scheduling considerations—to ascertain the likelihood of completing the project successfully. This assessment investigates whether any aspect of the proposed project conflicts with legal requirements like zoning laws, data protection acts or social media laws.

3.4.5 OPERATIONAL FEASIBILITY

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.



Fig. No. 3.2: Feasible study types

CHAPTER 4

EXPERIMENTAL MATERIALS AND METHODS, ALGORITHMS USED

4.1 SYSTEM ARCHITECTURE

4.1.1 MODULE DESCRIPTION

- DATA GATHERING
 - DATA PRE-PROCESSING
 - DATA TRAINING
 - DATA PREDECTION
-
- **Step1: DATA GATHERING**
 - Raw data gathered from internet.
 - **Step 2: DATA PRE-PROCESSING**
 - Cleaning of data removing the abnormalities and conflicts in the data.
 - **Step3: DATA TRAINING**
 - Applying training model with best efficiency.
 - **Step 4: DATA PREDECTION**
 - Determining the efficiency and predicting the output based on the threshold points initialized.

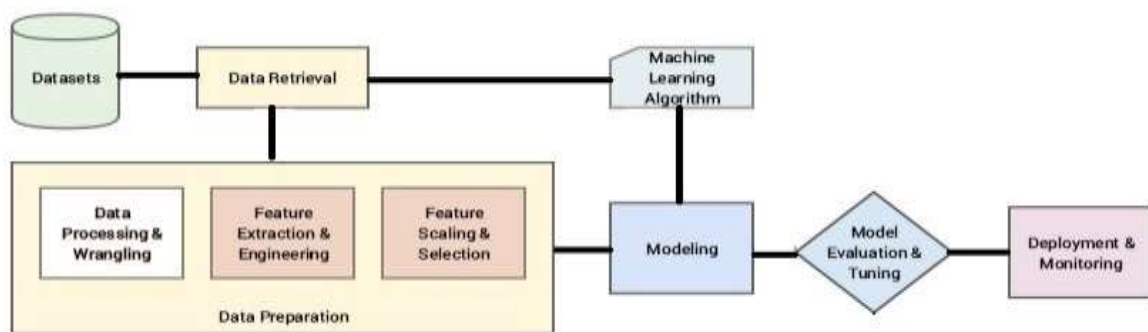


Fig. no. 4.1: System Architecture

4.2 ALGORITHM AND IMPLEMENTATION

4.2.1 COX REGRESSION

The algorithm is very simple. Cox regression (also known as proportional hazards regression) is a technique for determining the impact of multiple factors on the time it takes for an occurrence to occur. This is known as Cox regression for survival analysis in the form of an outcome such as death. We're dealing with right-censored data (it could also be left-censored, but we could assume time 0 is the verified date), so survival analysis seemed like a good fit in this situation. We'll develop and test our model's predictive power, as well as extract some key findings.

- Died: if died=1, the patient is diseased.
- Recovered: if recovered =1, the patient is recovered.
- Survived: The number of days that the patient has "survived" is the number of days that the patient has "survived." This is not to be confused with a patient's recovery. It refers to the number of days a patient has survived while being monitored. In this case, survival does not imply death or rehabilitation. As a result, if the patient is not dead or has not healed, the patient is censored, indicating that the outcome is unknown.

4.2.2 HAZARD AND BASELINE HAZARD

In Survival Analysis, Hazard is a function of time and in effect it models the probability of experiencing an event T given that the subject has survived more than t . In Cox Regression particularly, the hazard of a subject is described by the following function. In this exercise we use the hazard concept as the conditional probability of experiencing an event (death or recovery) given the patient's survival until a specific day. The features we are using for predicting the hazard are the age decade of the patient and the gender. But since we are operating in a competing risk setting, we will have to fit 2 different models. The first one outputs the baseline hazard for the death event, while the second one will output the baseline hazard of the recovery event. An important distinction that we need to make is that we censor competing events at the time of the occurrence. Let's say for example that we are

modelling the event of death. A patient who recovered 14 days after he was diagnosed, will be censored at 14 days of survival (the "survived" column in the dataset will have the value 14). The opposite will happen for a patient who died, when we will be modelling the event of recovery. This approach is the "cause specific" hazard approach, in contrast with "sub distribution" hazard approach, where in the latter, subjects that experience a competing risk remain in the risk set.

4.3 LIFELINE'S MODULE

Life lines are a fully functional survival research library written entirely in Python.

- Easy installation
- Internal plotting methods
- Simple and intuitive API
 - Handles right, left and interval censored data.
 - Contains most famous parameterized, non-parameterized and semi parameterized models.

4.4 SOFTWARE AND HARDWARE REQUIREMENTS

4.4.1 HARDWARE REQUIREMENTS

- System : Pentium Dual Core.
- Hard Disk : 120 GB.
- Monitor : 15" LED
- Input Devices : Keyboard, Mouse
- Ram : 1GB.

4.4.2 SOFTWARE REQUIREMENTS

- Operating system: Windows 10.
- Coding Language: Python
- Toolkit : Jupyter Notebook

CHAPTER 5

RESULTS AND DISCUSSIONS, PERFORMANCE ANALYSIS

5.1 RESULTS AND DISCUSSIONS

Thus, in the field of survival analysis, the latest generation of Cox-regression has achieved impressive results. This paper deals with a fresh approach to survival prediction of a person suffering with covid-19. In the medical sector, internal disease ailments and illness caused due to chronic damage of the tissues are a significant challenge. Faster and precise disease forecast in individuals and could assist creating an early therapy method while significantly lowering financial and human losses. Modern sophisticated machine learning innovations have enabled scientists to greatly enhance object detection and recognition systems efficiency and precision. After the successful execution of the code, we can get the desired rate of survival as output.

5.2 PERFORMANCE ANALYSIS

5.2.1 TEST PLAN

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also, to assess the feature of a software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words, software testing is a verification and validation process. Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to. Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

5.2.2 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and

those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

5.2.3 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making. The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

With the capacity to differentiate, this advanced model is able to acknowledge survival taking distinct kinds of parameters into consideration from previous covid patients and their symptoms history. All the essential steps required to implement this model of survival recognition are fully described throughout this paper, starting with the importing of packages, data reading, create a database based on the insights drawn after data segregation. Regression is the projects main method which includes picture acquisition, ROI adjustment, classification, function extraction and convolution. Lifeline's module python library and python programming language is used to manipulate the raw data and to create a machine learning model that can predict the type of disease.

6.2 FUTURE WORK

With technological development, the use of automated surveillance and management systems are growing in demand. The mechanism suggested can detect the disease previously as it happens. It is therefore feasible to save the loss and to reduce the reliance on the specialist to some extent. It can assist an individual with less understanding of the disease. We have to obtain the characteristics corresponding to the disease depending on these objectives. Using this system as framework a new concept of intelligent farming can be implemented where, using the self-operating systems, the field conditions are controlled and monitored.

REFERENCES

- [1] B. Saboonchi, P. Hansen, and S. Perron, “MaxMinMin p-dispersion problem: A variable neighborhood search approach,” *Comput. Oper. Res.*, vol. 52B, pp. 251–259, Dec. 2014.
- [2] C. Contardo, “Decremental clustering for the solution of p-dispersion problems to proven optimality,” *Inform. J. Optim.*, vol. 2, no. 2, pp. 134–144, Apr. 2020.
- [3] D. Sayah and S. Irnich, “A new compact formulation for the discrete p-dispersion problem,” *Eur. J. Oper. Res.*, vol. 256, no. 1, pp. 62–67, Jan. 2017.
- [4] D. Pisinger, “Upper bounds and exact algorithms for p-dispersion problems,” *Comput. Oper. Res.*, vol. 33, no. 5, pp. 1380–1398, May 2006.
- [5] E. Erkut, “The discrete p-dispersion problem,” *Eur. J. Oper. Res.*, vol. 46, no. 1, pp. 48–60, May 1990.
- [6] E. Erkut and S. Neuman, “Analytical models for locating undesirable facilities,” *Eur. J. Oper. Res.*, vol. 40, no. 3, pp. 275–291, Jun. 1989.
- [7] H. Al-Hiary, S. Bani-Ahmad, M. Reyalat, M. Braik and Z. ALRahamneh, Fast and Accurate Detection and Classification of Plant Diseases, *IJCA*, 2011, 17(1), 31-38, IEEE-2010.
- [8] I. D. Moon and S. S. Chaudhry, “An analysis of network location problems with distance constraints,” *Manage. Sci.*, vol. 30, no. 3, pp. 290–307, Mar. 1984.
- [9] Jayamala K. Patil, Raj Kumar, —Advances In Image Processing For Detection of Plant Diseases, *JABAR*, 2011, 2(2), 135-141.
- [10] M. M. Solomon, “Algorithms for the vehicle routing and scheduling problems with time window constraints,” *Oper. Res.*, vol. 35, no. 2, pp. 254–265, Apr. 1987.

- [11] M. J. Kuby, “Programming models for facility dispersion: The pdispersion and maxisum dispersion problems,” *Geograph. Anal.*, vol. 19, no. 4, pp. 315–329, Sep. 2010.
- [12] O. Braysy and M. Gendreau, “Vehicle routing problem with time windows—Part I: Route construction and local search algorithms,” *Transp. Sci.*, vol. 39, no. 1, pp. 104–118, Feb. 2005.
- [13] P.Revathi, M.Hemalatha, — *Classification of Cotton Leaf Spot Diseases Using Image Processing Edge Detection Techniques*, ISBN, 2012, 169-173, IEEE.
- [14] S. Arivazhagan, R. Newlin Shebiah, S. Ananthi, S. Vishnu Varthini, *Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features*, *Commission Internationale du Genie Rural(CIGR) journal*, vol.15, no.1, pp:211-217, March 2013.
- [15] V. Chamola, V. Hassija, V. Gupta, and M. Guizani, “A comprehensive review of the COVID-19 pandemic and the role of IoT, drones, AI, blockchain, and 5G in managing its impact,” *IEEE Access*, vol. 8, pp. 90225–90265, 2020.

APPENDIX

A. SOURCE CODE:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import warnings

import os
for dirname, _, filenames in os.walk('data'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
data = pd.read_csv('data/PatientInfo.csv')
timeProvince = pd.read_csv('data/TimeProvince.csv')
timeProvince.columns
time = pd.read_csv('data/Time.csv')
time.columns
timeAge = pd.read_csv('data/TimeAge.csv')
timeAge.columns
seoulFloating = pd.read_csv('data/SeoulFloating.csv')
seoulFloating.columns
! pip install lifelines
from datetime import datetime
from lifelines import CoxPHFitter
from lifelines.utils.sklearn_adapter import sklearn_adapter
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
import seaborn as sns
import numpy as np
import re
import math
sns.set(style="ticks", context="talk")
plt.style.use("dark_background")
pd.set_option('mode.chained_assignment', None)
data.columns
df = data
df['start_date'] = df.confirmed_date
df['start_date'] = pd.to_datetime(df['start_date'])
df['deceased_date'] = pd.to_datetime(df['deceased_date'])
df['released_date'] = pd.to_datetime(df['released_date'])
# The assumption here is that if the patient is not deceased or recovered, he is
censored, meaning we do not know the outcome yet.
df['survived_date'] =
df.deceased_date.fillna(df.released_date).fillna(datetime.today())

df = df.drop(df[(df.state == 'deceased') & (df.deceased_date.isna())].index)
df = df.drop(df[(df.state == 'released') & (df.released_date.isna())].index)

df['died'] = df.deceased_date.notnull().astype(int)
df['recovered'] = df.released_date.notnull().astype(int)
```

```

df['survived'] = (df.survived_date - df.start_date).dt.days

df['is_male'] = (df.sex == 'male').astype(int)
df['age_temp_a'] = df.age.apply(lambda x: int(re.findall("\d+", str(x))[0]) / 10 if
pd.notnull(x) else None)
df['age_temp_b'] = seoulFloating.birth_year.apply(lambda x: np.floor((2020 - x) /
10) if x != None else None)
df['age_decade'] = df.age_temp_a.fillna(df.age_temp_b)
df = df.drop(df[df.age_decade.isna()].index)
df = df[['patient_id', 'age_decade', 'is_male', 'died', 'recovered', 'survived']]
df = df.drop(df[df.isnull().any(axis = 1)].index)
df = df.drop(df[df.patient_id.duplicated()].index)
df.head()
train, test = train_test_split(df, test_size = 0.3)
X_train_death = train[['died', 'age_decade', 'is_male']]
y_train_death = train['survived']
X_test_death = test[['died', 'age_decade', 'is_male']]
y_test_death = test['survived']

X_train_recovery = train[['recovered', 'age_decade', 'is_male']]
y_train_recovery = train['survived']
X_test_recovery = test[['recovered', 'age_decade', 'is_male']]
y_test_recovery = test['survived']
print('train size : {}, test_size : {}'.format(train.shape[0], test.shape[0]))
def cox_ph_fitter(X_train, y_train, X_test, y_test, event):
    CoxRegression = sklearn_adapter(CoxPHFitter, event_col = event)
    cph = CoxRegression()
    cph.fit(X_train, y_train)
    cph.lifelines_model.print_summary()
    print('---')
    print('Test Score = {}'.format(cph.score(X_test, y_test)))
    return cph.lifelines_model
cph_death = cox_ph_fitter(X_train_death, y_train_death, X_test_death,
y_test_death, 'died')
cph_recovery = cox_ph_fitter(X_train_recovery, y_train_recovery,
X_test_recovery, y_test_recovery, 'recovered')
base_case_death = pd.Series({'age_decade' : 0, 'is_male' : 0})
cumulative_baseline_hazard_death =
cph_death.predict_cumulative_hazard(base_case_death)
baseline_hazard_death =
cumulative_baseline_hazard_death.diff().fillna(cumulative_baseline_hazard_d
eath)

base_case_recovery = pd.Series({'age_decade' : 0, 'is_male' : 0})
cumulative_baseline_hazard_recovery =
cph_recovery.predict_cumulative_hazard(base_case_recovery)
baseline_hazard_recovery =
cumulative_baseline_hazard_recovery.diff().fillna(cumulative_baseline_hazard
_recovery)

```

```

baseline_hazards = pd.merge(baseline_hazard_death[0:],
baseline_hazard_recovery[0:], left_index=True, right_index=True)
baseline_hazards = baseline_hazards.rename(columns = {'0_x' :
'baseline_hazard_death', '0_y' : 'baseline_hazard_recovery'})
baseline_hazards.head()
test['key'] = 1
baseline_hazards['key'] = 1

#cross join of patient in the test set with the baseline hazards
test_lines = pd.merge(test, baseline_hazards[:30], on = 'key')

#simple row number, indicating the number of days since confirmed.
test_lines['days'] = test_lines.groupby('patient_id').cumcount()

#according to the definition of the hazard function in cox regression, we calculate
the hazard for each patient individually
#in effect this is the conditional probability of death and recovery for each patient,
given the patient has survived so far
test_lines['p_death_given_s'] = test_lines.baseline_hazard_death *
np.exp(np.dot(test_lines[['age_decade', 'is_male']], cph_death.params_))
test_lines['p_recovery_given_s'] = test_lines.baseline_hazard_recovery *
np.exp(np.dot(test_lines[['age_decade', 'is_male']], cph_recovery.params_))

test_lines['survive_death'] = 1-test_lines.p_death_given_s
test_lines['survive_recovery'] = 1-test_lines.p_recovery_given_s

#probability of survival defined as if the patient did not die and did not recover so
far
test_lines['p_survive'] = test_lines.groupby('patient_id').survive_death.cumprod() *
test_lines.groupby('patient_id').survive_recovery.cumprod()

#unconditional probability of death and recovery.
test_lines['p_death'] = test_lines.p_death_given_s *
test_lines.groupby('patient_id').p_survive.shift().fillna(1)
test_lines['p_recovery'] = test_lines.p_recovery_given_s *
test_lines.groupby('patient_id').p_survive.shift().fillna(1)

#observed deaths and recoveries
test_lines['recovered'] = ((test_lines.survived == test_lines.days) &
(test_lines.recovered == 1)).astype(int)
test_lines['died'] = ((test_lines.survived == test_lines.days) & (test_lines.died ==
1)).astype(int)
warnings.filterwarnings("ignore")
agg = test_lines.groupby('days').agg({
    'died' : np.sum
    , 'recovered' : np.sum
    , 'p_death' : np.sum
    , 'p_recovery' : np.sum
    , 'patient_id' : pd.Series.nunique
}).reset_index()

```

```

agg['death_rate'] = agg.died.cumsum() / agg.patient_id
agg['death_rate_pred'] = agg.p_death.cumsum() / agg.patient_id
agg['recovery_rate'] = agg.recovered.cumsum() / agg.patient_id
agg['recovery_rate_pred'] = agg.p_recovery.cumsum() / agg.patient_id
warnings.filterwarnings("ignore")
sns.lineplot(agg.days, agg.death_rate_pred, label = 'predictions')
sns.lineplot(agg.days, agg.death_rate, label = 'actuals')
sns.lineplot(agg.days, agg.recovery_rate_pred, label = 'predictions')
sns.lineplot(agg.days, agg.recovery_rate, label = 'actuals')
warnings.filterwarnings("ignore")
def cox_ph_fitter(X_train, y_train, X_test, y_test, event):
    CoxRegression = sklearn_adapter(CoxPHFitter, event_col = event)
    cph = CoxRegression()
    cph.fit(X_train, y_train)
    #cph.lifelines_model.print_summary()
    #print('---')
    #print('Test Score = {}'.format(cph.score(X_test, y_test)))
    return cph.lifelines_model

aggregations = pd.DataFrame(columns = ['iteration', 'days', 'death_rate',
'death_rate_pred', 'recovery_rate', 'recovery_rate_pred'])

for iteration in range (1,21):
    train, test = train_test_split(df, test_size = 0.3)
    X_train_death = train[['died', 'age_decade', 'is_male']]
    y_train_death = train['survived']
    X_test_death = test[['died', 'age_decade', 'is_male']]
    y_test_death = test['survived']

    X_train_recovery = train[['recovered', 'age_decade', 'is_male']]
    y_train_recovery = train['survived']
    X_test_recovery = test[['recovered', 'age_decade', 'is_male']]
    y_test_recovery = test['survived']

    cph_death = cox_ph_fitter(X_train_death, y_train_death, X_test_death,
y_test_death, 'died')
    cph_recovery = cox_ph_fitter(X_train_recovery, y_train_recovery,
X_test_recovery, y_test_recovery, 'recovered')

    base_case_death = pd.Series({'age_decade' : 0, 'is_male' : 0})
    cummulative_baseline_hazard_death =
cph_death.predict_cumulative_hazard(base_case_death)
    baseline_hazard_death =
cummulative_baseline_hazard_death.diff().fillna(cummulative_baseline_hazard_d
eath)

    base_case_recovery = pd.Series({'age_decade' : 0, 'is_male' : 0})
    cummulative_baseline_hazard_recovery =
cph_recovery.predict_cumulative_hazard(base_case_recovery)
    baseline_hazard_recovery =

```

```
cummulative_baseline_hazard_recovery.diff()).fillna(cummulative_baseline_hazard_recovery)
```

```
baseline_hazards = pd.merge(baseline_hazard_death[0:],
baseline_hazard_recovery[0:], left_index=True, right_index=True)
baseline_hazards = baseline_hazards.rename(columns = {'0_x' :
'baseline_hazard_death', '0_y' : 'baseline_hazard_recovery'})
```

```
test['key'] = 1
baseline_hazards['key'] = 1
```

```
#cross join of patient in the test set with the baseline hazards
test_lines = pd.merge(test, baseline_hazards[:30], on = 'key')
```

```
#simple row number, indicating the number of days since confirmed.
test_lines['days'] = test_lines.groupby('patient_id').cumcount()
```

```
#according to the definition of the hazard function in cox regression, we
calculate the hazard for each patient individually
```

```
#in effect this is the conditional probability of death and recovery for each
patient, given the patient has survived so far
```

```
test_lines['p_death_given_s'] = test_lines.baseline_hazard_death *
np.exp(np.dot(test_lines[['age_decade', 'is_male']], cph_death.params_))
test_lines['p_recovery_given_s'] = test_lines.baseline_hazard_recovery *
np.exp(np.dot(test_lines[['age_decade', 'is_male']], cph_recovery.params_))
```

```
test_lines['survive_death'] = 1-test_lines.p_death_given_s
test_lines['survive_recovery'] = 1-test_lines.p_recovery_given_s
```

```
#probability of survival defined as if the patient did not die and did not recover
so far
```

```
test_lines['p_survive'] = test_lines.groupby('patient_id').survive_death.cumprod()
* test_lines.groupby('patient_id').survive_recovery.cumprod()
```

```
#unconditional probability of death and recovery.
```

```
test_lines['p_death'] = test_lines.p_death_given_s *
test_lines.groupby('patient_id').p_survive.shift().fillna(1)
test_lines['p_recovery'] = test_lines.p_recovery_given_s *
test_lines.groupby('patient_id').p_survive.shift().fillna(1)
```

```
#observed deaths and recoveries
```

```
test_lines['recovered'] = ((test_lines.survived == test_lines.days) &
(test_lines.recovered == 1)).astype(int)
test_lines['died'] = ((test_lines.survived == test_lines.days) & (test_lines.died ==
1)).astype(int)
```

```
agg = test_lines.groupby('days').agg({
'died' : np.sum
'recovered' : np.sum
'p_death' : np.sum
```

```

    , 'p_recovery' : np.sum
    , 'patient_id' : pd.Series.nunique
}).reset_index()
agg['death_rate'] = agg.died.cumsum() / agg.patient_id
agg['death_rate_pred'] = agg.p_death.cumsum() / agg.patient_id
agg['recovery_rate'] = agg.recovered.cumsum() / agg.patient_id
agg['recovery_rate_pred'] = agg.p_recovery.cumsum() / agg.patient_id

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12,4))
sns.lineplot(agg.days, agg.death_rate_pred, label = 'predictions', ax = ax1)
sns.lineplot(agg.days, agg.death_rate, label = 'actuals', ax = ax1)

sns.lineplot(agg.days, agg.recovery_rate_pred, label = 'predictions', ax = ax2)
sns.lineplot(agg.days, agg.recovery_rate, label = 'actuals', ax = ax2)
fig.suptitle('Iteration : {}'.format(iteration) )

agg['iteration'] = iteration
agg_to_append = agg[['iteration', 'days', 'death_rate', 'death_rate_pred',
'recovery_rate', 'recovery_rate_pred']]
aggregations = aggregations.append(agg_to_append, ignore_index = True)

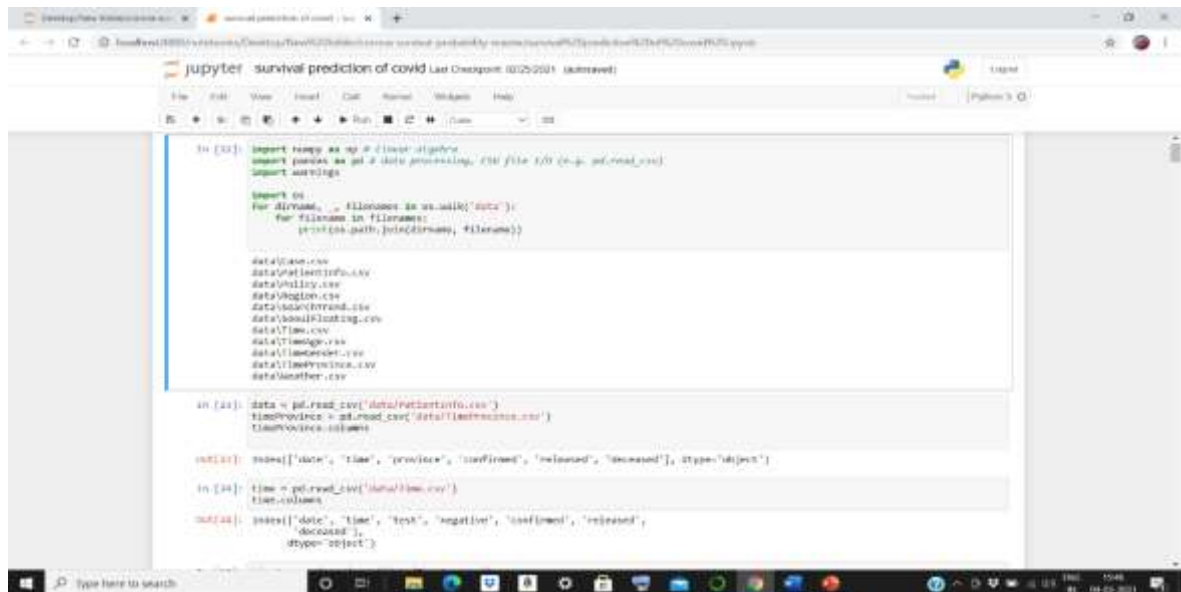
warnings.filterwarnings("ignore")
aggregations['squared_death_diff'] = (aggregations.death_rate -
aggregations.death_rate_pred) ** 2
aggregations['squared_recovery_diff'] = (aggregations.recovery_rate -
aggregations.recovery_rate_pred) ** 2
errors = aggregations.groupby('days').agg({
    'squared_death_diff' : np.sum
    , 'squared_recovery_diff' : np.sum
    , 'iteration' : np.size
}).reset_index()
errors['RMSD_death_rate'] = np.sqrt(errors.squared_death_diff / errors.iteration)
errors['RMSD_recovery_rate'] = np.sqrt(errors.squared_recovery_diff /
errors.iteration)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16,6))
sns.lineplot(errors.days, errors.RMSD_death_rate, ax = ax1)
sns.lineplot(errors.days, errors.RMSD_recovery_rate, ax = ax2)

```

B. SCREEN SHOTS:

PYTHON CODE:



```
import numpy as np
import pandas as pd
import os
import warnings

import sys
for dirname, _, filenames in os.walk('data'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

data/date.csv
data/recoveryinfo.csv
data/holiday.csv
data/holiday.csv
data/holiday.csv
data/holiday.csv
data/holiday.csv
data/holiday.csv
data/holiday.csv
data/holiday.csv
data/holiday.csv

In [21]: data = pd.read_csv('data/recoveryinfo.csv')
timeProvince = pd.read_csv('data/timeProvince.csv')
timeProvince.columns

Out[21]: Index(['date', 'time', 'province', 'confirmed', 'released', 'deceased'], dtype='object')

In [22]: time = pd.read_csv('data/time.csv')
time.columns

Out[22]: Index(['date', 'time', 'test', 'negative', 'confirmed', 'released', 'deceased'], dtype='object')
```

Fig.no. B.1: python code

We have to run the code in jupyter note book to get the required output. The first steps include importing packages, data reading, printing data columns and data segregation.

APPLICATION OF COX REGRESSION:

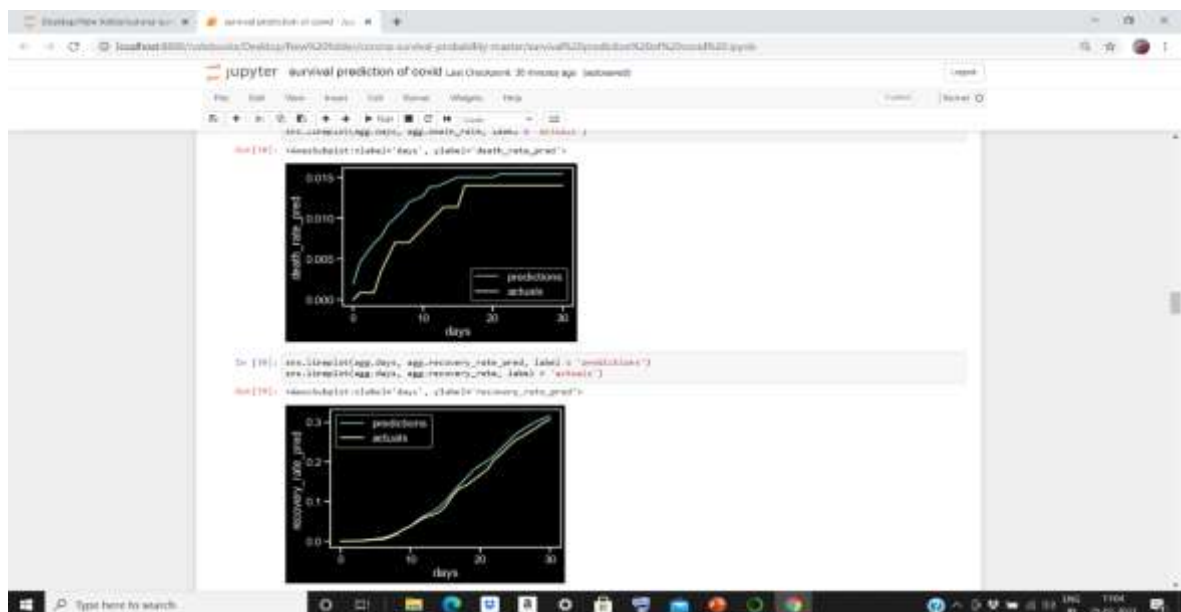


Fig.no. B.2: Cox Regression in both the cases

After data segregation, we need to apply cox regression in both cases i.e for died as input and for recovered as input. With given died and recovered insights we can predict death prediction rate as well as recovered prediction rate which in turn helps us to attain at survival prediction rate.

EPOCH TECHNIQUE:

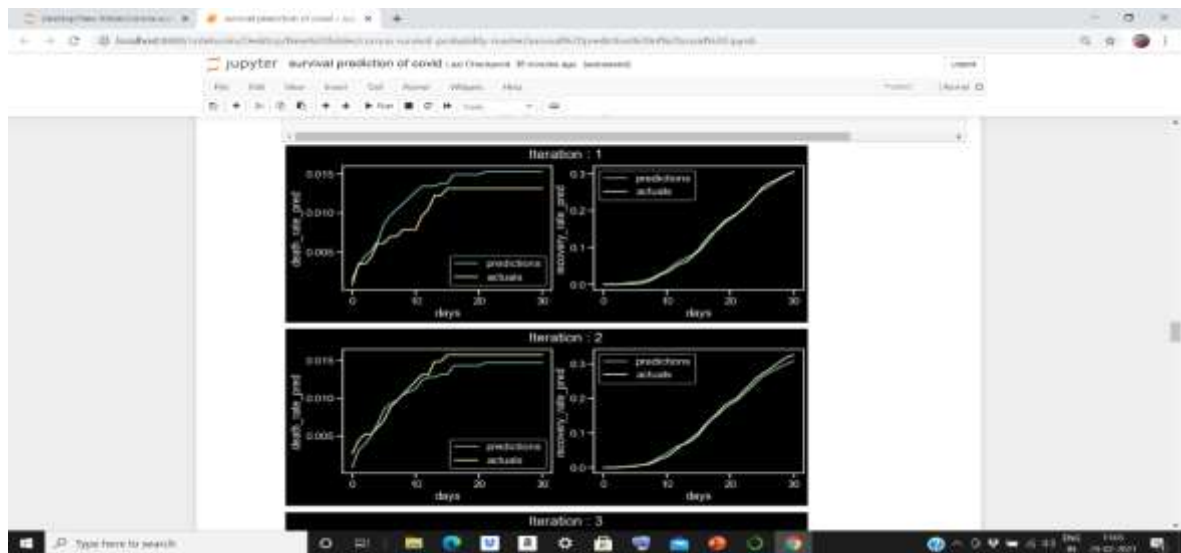


Fig.no. B.3: Application of epoch technique

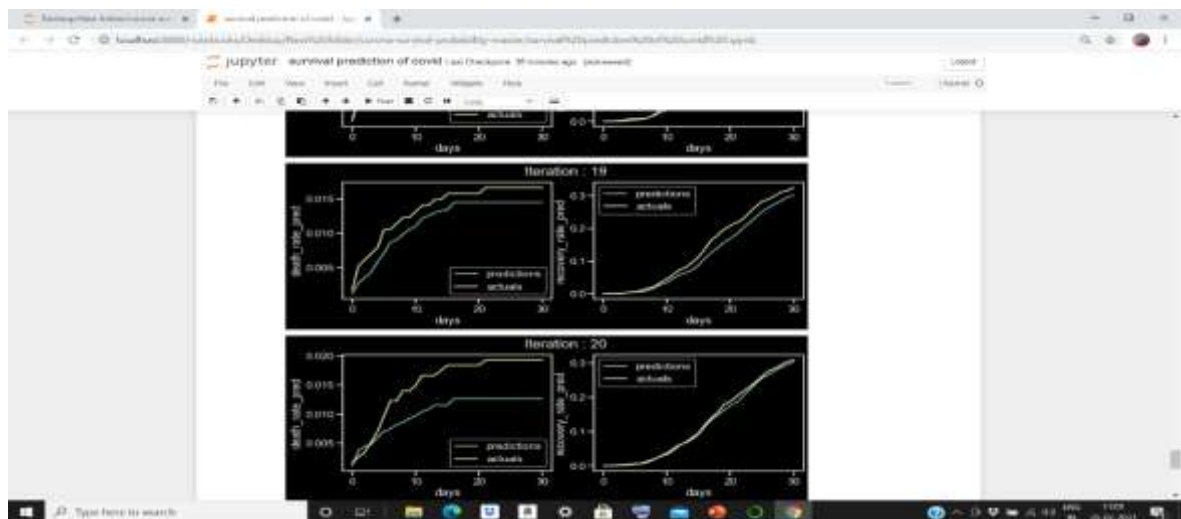


Fig.no. B.4: Epoch technique -20 iterations

In order to increase the performance of the algorithm, we need to apply epoch technique for cross validation and training the data in an iterative manner.

SURVIVAL RATE PREDICTION:

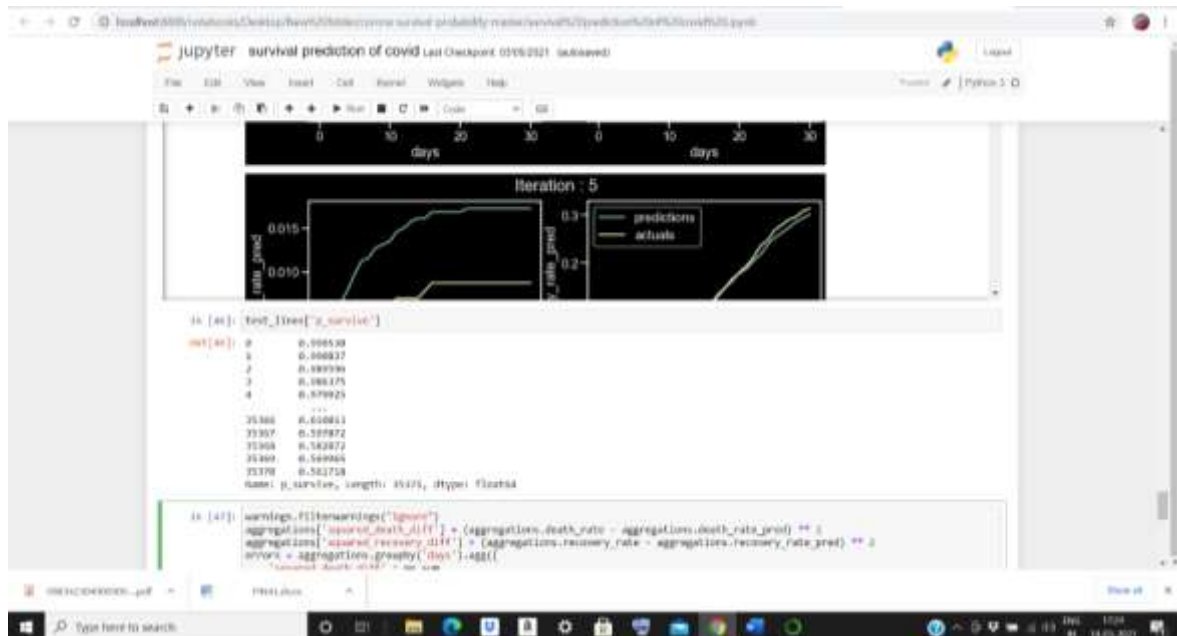


Fig.no. B.5: Survival prediction rates

In a similar way, we can predict the survival rates for each and every possible input.

RMSD ERROR:

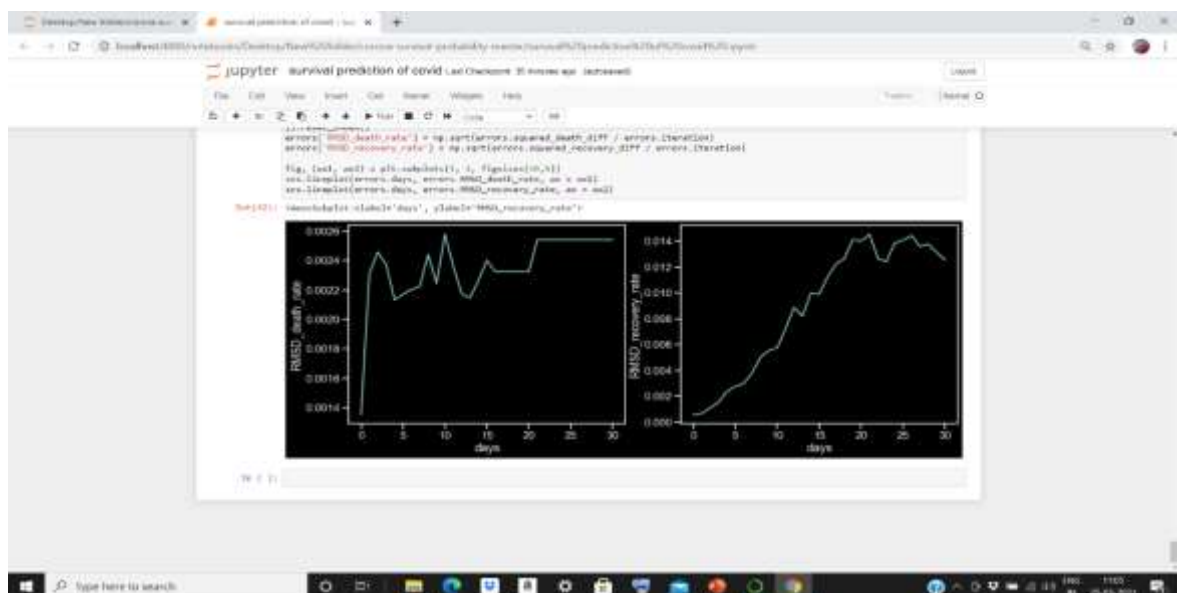


Fig.no. B.6: RMSD Curve

To compute the errors and visualize them we have predicted the RMSD curves in both the cases.