

COVID19 SURVIVAL PREDICTION

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

by

CHILUVURI SAI HEMANTH VIHARI

(Reg. No.37110147)

CHINTAKUNTLA SAI CHARAN REDDY

(Reg. No. 37110151)



**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
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600 119**

MARCH - 2021



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 **CHILUVURI SAI HEMANTH VIHARI (Reg. No. 37110147)** and **CHINTAKUNTLA SAI CHARAN REDDY (Reg. No.37110151)** who carried out the project entitled "**COVID19 SURVIVAL PREDICTION**" under my supervision from **August 2020** to **March 2021**.

Internal Guide

Dr.Indira,M.E.,Ph.d.,

Head of the Department

Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **CHILUVURI SAI HEMANTH VIHARI** hereby declare that the Project Report entitled “**COVID19 SURVIVAL PREDICTION**” is done by me under the guidance of **Dr.Indira,M.E.,Ph.d.**, Department of Computer Science and Engineering 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, **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. Indira, M.E., Ph.D., Assistant Professor**, 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

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. Many countries have imposed social distancing measures that require a minimum distance between people in given places, such as schools, restaurants, shops, etc. This in turn creates complications for these places, as their function is to serve as many people as they were originally designed for. In this article, we pose the problem of using the available space in a given place, such that the social distancing measures are satisfied, as a p-dispersion problem. We use recent algorithmic advancements, that were developed for the p-dispersion problem, and combine them with discretization schemes to find computationally attainable solutions to the p-dispersion problem and investigate the trade-off between the level of discretization and computational efforts on one side, and the value of the optimal solution on the other.

TABLE OF CONTENTS

	ABSTRACT	v
		viii
	LIST OF FIGURES	
CHAPTER No.	TITLE	PAGE No.
1.	INTRODUCTION	1
	1.1 INTRODUCTION	1
2.	LITERATURE SURVEY	3
	2.1 SEGMENTATION AND CLASSIFICATION OF LESIONS FOR DISEASE DIAGNOSIS	3
	2.2 EXTRACTION OF LESIONS FROM NON- DERMOSCOPIC IMAGES USING DEEP LEARNING	3
	2.3 AN INTELLIGENT SYSTEM TO DIAGNOSIS THE DISEASE	4
	2.4 AUTOMATIC SEGMENTATION OF LESIONS FROM DERMATOLOGICAL PHOTOGRAPH	4
	2.5 A SURVEY ON SKIN TEXTURE ANALYSIS FOR MEDICAL DIAGNOSIS USING IMAGE PROCESSING TECHNIQUES	5
		6
3.	AIM AND SCOPE	
	3.1 AIM	6
	3.2 OBJECTIVE	6
	3.3 SCOPE	6
4.	SYSTEM DESIGN, METHODOLOGY AND ALGORITHMS USED	7
	4.1 EXISTING SYSTEM	7
	4.2 PROPOSRD SYSTEM	7
	4.2.1 MODULUD DESCRIPTION	8
	4.3 DATA PREPROCESSING AND TRANSFER LEARNING	8
	4.7 SYSTEM REQUIREMENTS	11
	4.7.1 HARDWARE REQUIREMENTS	11
	4.7.2 SOFTWARE REQUIREMENTS	11

4.8	REQUIREMENT ANALYSIS	12
4.8.1	FUNCTIONAL REQUIREMENTS	12
4.8.2	NON-FUNCTIONAL REQUIREMENTS	13
4.9	FEASIBILITY STUDY	14
5.	PERFORMANCE ANALYSIS, DISCUSSION AND RESULT	27
5.3	PURPOSE	27
5.5	SURVIVAL ANALYSIS-COX REGRESSION	28
5.6	HAZARD AND BASELINE HAZARD	28
5.7	CROSS VALIDATION	31
5.8	RESULT	38
6.	CONCLUSION	39
7.	REFERENCES	40
8.	APPENDIX	41
8.1	SOURCE CODE	41

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
4.4	ARCHITECTURE DIAGRAM	9
4.5	DATA FLOW DIAGRAM	10
4.6	SYSTEM ARCHITECTURE	11
5.1	PREPROCESSING TRAINING DATASET	27
5.2	PREPROCESSING TESTING DATASET	27
5.4	BUILDING THE DATASET	28

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The outbreak of the COVID-19 had an enormous impact on the world at large. To mitigate the spread of the virus, various technologies, such as Internet of Things, Unmanned Aerial Vehicles, block chain, Artificial Intelligence, and 5G are already in use [1]. In this article, we take a look at the problem of positioning people in a given area, such as in a restaurant, school, office, etc., in order to minimize the spread of viruses such as COVID-19. After the initial lockdown, many countries imposed a set of social distancing measures that should help to slow down the spread of the virus. These measures impose a minimum distance between people in a given area. This means that spaces that could previously serve a large number people need to be adjusted for these new measures. As it seems unlikely that we will see the construction of new places that will be designed to abide by these (hopefully temporary) measures, it is only natural to try to find the best use of the “facilities” that are already available. However, as the social distancing measures do not have to be stable and can change over time, we will pose the problem of using the available space to its full extent in the following way: Given a fixed number p of people, fit them into a predefined space in such a way, that the minimum distance between any two persons is maximized. Afterwards, by varying p , we can get the optimal (largest) distance that the people can be separated by, and, given a particular social distancing rule, we can determine the maximum number (and placement) of people that will fit into the predefined space. The problem of selecting p points in order to maximize the minimum distance between any pair is called the p -dispersion problem [2] and it is one of the classical combinatorial optimization problems. Although easy to formulate, effective and provably optimal methods for solving this problem are quite a recent development. Most notably, the state-of-the-art methods are based on the formulation developed in 2017 in [3] and the most successful method is the decremented clustering scheme published in 2020 in [4]. It is these advancements that made it possible to solve instances of the size sufficient for our purpose. In this article, we devise a discretization scheme that is build on top of the decremented clustering to find computationally attainable solutions to the p -dispersion problem and investigate the trade-off between the level of discretization and computational efforts on

one side, and the value of the optimal solution (the minimum distance between any two points) on the other. The investigation is carried out on two numerical examples, the first one is a place with a “general” shape, the second one with an “auditorium-like” shape. In this article, we pose the problem of using the available space in a given place, such that the social distancing measures are satisfied, as a p-dispersion problem. We use recent algorithmic advancements, that were developed for the p-dispersion problem, and combine them with discretization schemes to find computationally attainable solutions to the p-dispersion problem and investigate the trade-off between the level of discretization and computational efforts on one side, and the value of the optimal solution on the other.

CHAPTER 2

LITERATURE SURVEY

2.1 SEGMENTATION AND CLASSIFICATION OF LESIONS FOR DISEASE DIAGNOSIS

In this paper, a novel approach for automatic segmentation and classification of skin lesions is proposed. Initially, skin images are filtered to remove unwanted hairs and noise and then the segmentation process is carried out to extract lesion areas. For segmentation, a region growing method is applied by automatic initialization of seed points. The segmentation performance is measured with different well known measures and the results are appreciable. Subsequently, the extracted lesion areas are represented by color and texture features. SVM and k-NN classifiers are used along with their fusion for the classification using the extracted features. The performance of the system is tested on our own dataset of 726 samples from 141 images consisting of 5 different classes of diseases. The results are very promising with 46.71% and 34% of F-measure using SVM and k-NN classifier respectively and with 61% of F-measure for fusion of SVM and k-NN.

2.2 EXTRACTION OF LESIONS FROM NON-DERMOSCOPIC IMAGES USING DEEP LEARNING

Melanoma is amongst most aggressive types of cancer. However, it is highly curable if detected in its early stages. Prescreening of suspicious moles and lesions for malignancy is of great importance. Detection can be done by images captured by standard cameras, which are more preferable due to low cost and availability. One important step in computerized evaluation of skin lesions is accurate detection of lesion region, i.e. segmentation of an image into two regions as lesion and normal skin. Accurate segmentation can be challenging due to burdens such as illumination variation and low contrast between lesion and healthy skin. In this paper, a method based on deep neural networks is proposed for accurate extraction of a lesion region. The input image is preprocessed and then its patches are fed to a convolution neural network. Local texture and global structure of the patches are processed in order to assign pixels to lesion or normal classes. A method for effective selection of training patches is used for more accurate detection of a lesion border. The output segmentation mask is refined by some post processing operations. The experimental results of qualitative and quantitative evaluations demonstrate that our method can outperform other state-of-the-art algorithms exist in the literature.

2.3 AN INTELLIGENT SYSTEM TO DIAGNOSIS THE DISEASE

Skin cancer is a major health issue affecting a vast segment of the population regardless the skin color. This affectation can be detected using dermoscopy to determine whether the visible spots on skin are either benign or malignant tumors. In spite of the specialists' experience, skin lesions are difficult to classify, reason for which computer systems are developed to increase the effectiveness of cancer detection. Systems assisting in the detection of skin cancer process digital images to determine the occurrence of tumors by interpreting clinical parameters, relying, firstly, upon an accurate segmentation process to extract relevant features. Two of the well-known methods to analyze lesions are ABCD and the 7-point check list. After clinically-relevant features are extracted, they are used to classify the presence or absence of a tumor. However, irregular and disperse lesion borders, low contrast, artifacts in images and the presence of various colors within the region of interest complicate the processing of images. In this article, we propose an intelligent system running the following method. The feature extraction stage begins with the segmentation of an image, for which we apply the Wavelet – Fuzzy C-Means algorithm. Next, specific features should be determined, among others the area and the asymmetry of the lesion. An ensemble of clusterers extracts the Red-Green-Blue values that correspond to one or more of the colors defined in the ABCD guide. The feature extraction stage includes the discovery of structures that appear in the lesion according to the method known as Grey Level Co Occurrence Matrix. Then, during the detection phase, an ensemble of classifiers determines the occurrence of a malignant tumor. Our experiments are performed on images taken from the ISIC repository. The proposed system provides a skin cancer detection performance above 88 percent, as measured by the accuracy. Details of how this performance fares when compared with other systems are also given.

2.4 AUTOMATIC SEGMENTATION OF LESIONS FROM DERMATOLOGICAL PHOTOGRAPH

Melanoma is the deadliest form of skin cancer if left untreated. Incidence rates of melanoma have been increasing, especially among young adults, but survival rates are high if detected early. Unfortunately, the time and costs required for dermatologists to screen all patients for melanoma are prohibitively expensive. There is a need for an automated system to assess a patient's risk of melanoma using photographs of their skin lesions. Dermatologists could use the system to aid their diagnosis without the need for special or expensive equipment. One challenge in implementing such a system is locating

the skin lesion in the digital image. Most existing skin lesion segmentation algorithms are designed for images taken using a special instrument called the dermatoscope. The presence of illumination variation in digital images such as shadows complicates the task of finding the lesion. The goal of this research is to develop a framework to automatically correct and segment the skin lesion from an input photograph. The first part of the research is to model illumination variation using a proposed multi-stage illumination modeling algorithm and then using that model to correct the original photograph. Second, a set of representative texture distributions are learned from the corrected photograph and a texture distinctiveness metric is calculated for each distribution. Finally, a texture-based segmentation algorithm classifies regions in the photograph as normal skin or lesion based on the occurrence of representative texture distributions. The resulting segmentation can be used as an input to separate feature extraction and melanoma classification algorithms. The proposed segmentation framework is tested by comparing lesion segmentation results and melanoma classification results to results using other state-of-the-art algorithms. The proposed framework has better segmentation accuracy compared to all other tested algorithms. The segmentation results produced by the tested algorithms are used to train an existing classification algorithm to identify lesions as melanoma or non-melanoma. Using the proposed framework produces the highest classification accuracy and is tied for the highest sensitivity and specificity.

2.5 A SURVEY ON SKIN TEXTURE ANALYSIS FOR MEDICAL DIAGNOSIS USING IMAGE PROCESSING TECHNIQUES

Skin texture analysis is one of them challenging issues in the field of medical diagnosis. Various types of skin diseases are affecting human life like skin dryness, fungus, and allergic symptoms. The objective of this paper is to analyze the skin disease using texture analysis of skin image and by comparing the test image to a defined images or reference images. The matching of test and reference images compared that yields the percentage of skin diseases in the captured skin texture image.

CHAPTER 3

AIM AND SCOPE

3.1 AIM

The main aim of this system is early disease recognition, patient care and community services. When the quality of medical data is incomplete the exactness of study is reduced.

3.2 OBJECTIVE

The analysis accuracy is reduced when the quality of medical data is incomplete. Moreover, different regions exhibit unique characteristics of certain regional diseases, which may weaken the prediction of disease outbreaks. However, those existing works mostly considered structured data. There is no proper method to handle semi structured and unstructured data. The proposed system will consider both structured and unstructured data. The analysis accuracy is increased by using Machine Learning algorithm.

3.3 SCOPE

In this work, our goal is to provide a tool to assist professionals and consumers in finding and choosing disease. To achieve this goal, we develop an approach that allows a user to query for disease that satisfies a set of conditions based on disease properties, such as disease indications.

CHAPTER 4

SYSTEM DESIGN, METHODOLOGY AND ALGORITHMS USED

4.1 EXISTING SYSTEM

We researched the diagnostic capabilities of deep learning on chest radiographs and an image classifier based on the COVID-Net was presented to classify chest X-Ray images. In the case of a small amount of COVID-19 data, data enhancement was proposed to expand COVID-19 data 17 times. Our model aims at transfer learning, model integration and classifies chest X ray images according to three labels: normal, COVID-19 and viral pneumonia. According to the accuracy and loss value, choose the models ResNet-101 and ResNet-152 with good effect for fusion, and dynamically improve their weight ratio during the training process.

4.2 PROPOSED SYSTEM

This paper concerns a class of network location problems with minimum or maximum separation requirements between (incapacitated) facilities or between demand points and the facilities, or both. Its purpose is three-fold. First, it is to recognize distance constraints as increasing real life restrictions through various motivating illustrations. Using a new classification scheme, the paper introduces a variety of distance-constrained problems defined in a unified manner. These include a number of new problems. Second, it is to survey existing solution techniques, available only for a few of such constrained problems. 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. In particular, the paper presents integer programming formulations of several new problems along with the results of applying linear programming relaxation methods. Although the computational experience is somewhat disappointing for some of these problems, the results provide greater insight into the problems. With the stated purpose, it is hoped that this paper will stimulate future research in this important problem area.

4.2.1 MODULUS DESCRIPTION

- Tracking COVID-19
- Detecting COVID-19
- Location based COVID health suggestions
- Data Preprocessing
- Transfer Learning

4.2.2 TRACKING COVID-19

The trouble is, all of these proposed digital solutions are flawed, some fundamentally so. GPS tracking, which records location data on a shared database, is fraught with privacy concerns. And that could lead to legal challenges both in the U.S. and abroad. The technology is also imperfect.

4.2.3 DETECTING COVID-19

Detecting COVID-19 early may help in devising an appropriate treatment plan and disease containment decisions. In this study, we demonstrate how transfer learning from deep learning models can be used to perform COVID-19 detection using images from three most commonly used medical imaging modes X-Ray, Ultrasound, and CT scan. The aim is to provide over-stressed medical professionals a second pair of eyes through intelligent deep learning image classification models.

4.2.4 LOCATION BASED COVID HEALTH SUGGESTIONS

A single app, or wearable, isn't going to be what saves us. Contact tracing apps and symptom trackers may help us contain the infection, said Chan, but in the end, they need to be supported by a solid foundation of expanded testing, a well-equipped health care system prepared to handle an outbreak, and ultimately, a vaccine.

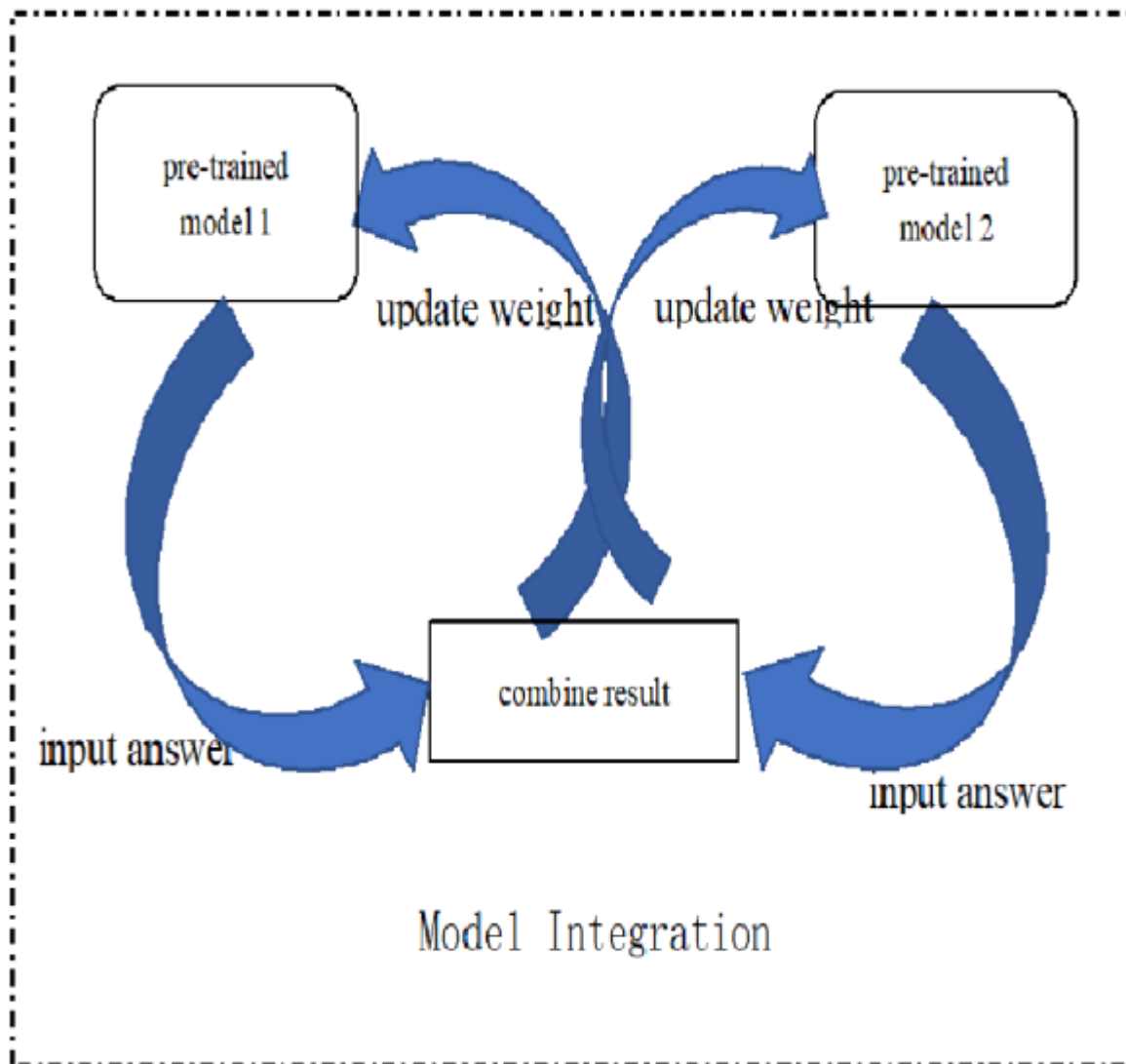
4.3 DATA PREPROCESSING AND TRANSFER LEARNING

Loading data to solve unbalanced In order to verify the feasibility of the method proposed in this paper, a representative data set that was open source during the COVID-19 epidemic was searched.

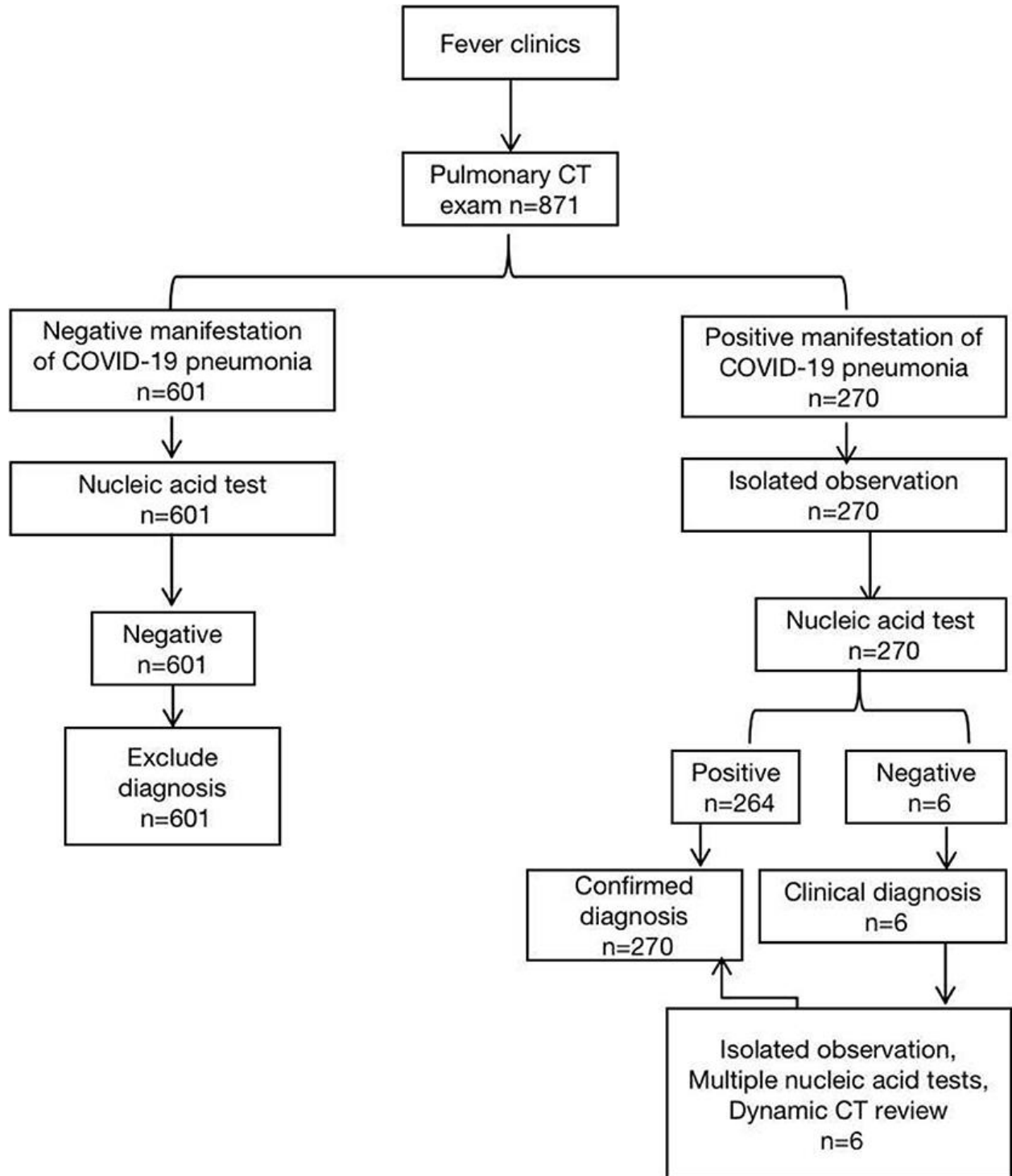
4.3.1 TRANSFER LEARNING

The deeper the neural network, the more data training is needed to better fit the parameters of each layer. Otherwise, it is particularly easy to over fit, resulting in particularly poor generalization ability. In response to this problem, transfer learning provides a feasible solution. It makes full use of the generalization ability of training models on large data sets.

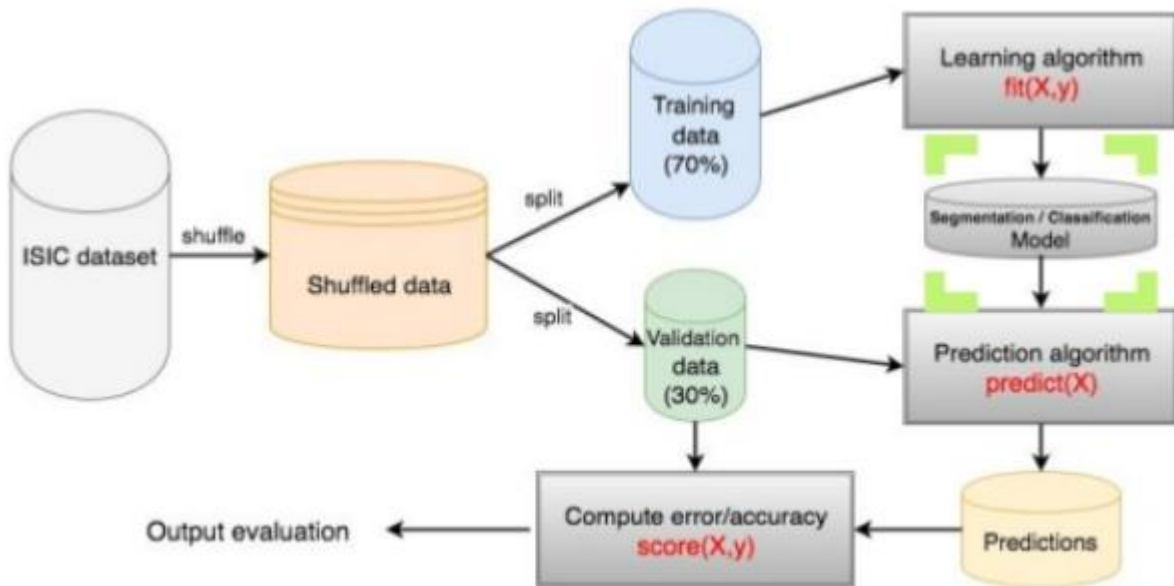
4.4 ARCHITECTURE DIAGRAM



4.5 DATA FLOW DIAGRAM



4.6 SYSTEM ARCHITECTURE



4.7 SYSTEM REQUIREMENTS

4.7.1 HARDWARE REQUIREMENTS

System	: Pentium Dual Core.
Hard Disk	: 120 GB.
Monitor	: 15"LED
Input Devices	: Keyboard, Mouse
Ram	: 4GB.

4.7.2 SOFTWARE REQUIREMENTS

Operating system	: Windows10.
Coding Language	: Python
IDE	: jupyter
Database	: mySQL

4.8 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.

4.8.1 FUNCTIONAL REQUIREMENTS

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.

4.8.2 NON- FUNCTIONAL REQUIREMENTS**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.

4.9 FEASIBILITY 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.

4.9.1 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 justified. 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.

4.9.2 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.

4.9.3 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.

4.9.4 SYSTEM DESIGN AND TESTING PLAN

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.

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.

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

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

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.

Basics of software testing

There are two basics of software testing: black box testing and white box testing.

Black box Testing

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.

White box Testing

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called structural testing and glass box testing. Black box

testing is often used for validation and white box testing is often used for verification.

Types of testing

There are many types of testing like

- Unit Testing
- Integration Testing
- Functional Testing
- System Testing
- Stress Testing
- Performance Testing
- Usability Testing
- Acceptance Testing
- Regression Testing
- Beta Testing

Unit Testing

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

Integration Testing

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

Functional Testing

Functional testing is the testing to ensure that the specified functionality required in the system requirements works. It falls under the class of black box testing.

System Testing

System testing is the testing to ensure that by putting the software in different environments

(e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing.

Stress Testing

Stress testing is the testing to evaluate how system behaves under unfavorable conditions. Testing is conducted at beyond limits of the specifications. It falls under the class of black box testing.

Performance Testing

Performance testing is the testing to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements. It falls under the class of black box testing.

Usability Testing

Usability testing is performed to the perspective of the client, to evaluate how the GUI is user-friendly? How easily can the client learn? After learning how to use, how proficiently can the client perform? How pleasing is it to use its design? This falls under the class of black box testing.

Acceptance Testing

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the customer expected. It falls under the class of black box testing.

Regression Testing

Regression testing is the testing after modification of a system, component, or a group of related units to ensure that the modification is working correctly and is not damaging or imposing other modules to produce unexpected results. It falls under the class of black box testing.

4.9.5 SOFTWARE ENVIRONMENT

PYTHON:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is interpreted** – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and UNIX shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

PYTHON FEATURES

Python's features include –

- **Easy-to-learn** – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** – Python code is more clearly defined and visible to the eyes.

- **Easy-to-maintain** – Python's source code is fairly easy-to-maintain.
- **A broad standard library** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** – Python provides interfaces to all major commercial databases.
- **GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** – Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

GETTING PYTHON

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python <https://www.python.org>.

Windows Installation

Here are the steps to install Python on Windows machine.

- Open a Web browser and go to <https://www.python.org/downloads/>.
- Follow the link for the Windows installer python-XYZ.msifile where XYZ is the version you need to install.
- To use this installer python-XYZ.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

The Python language has many similarities to Perl, C, and Java. However, there are some definite differences between the languages.

FIRST PYTHON PROGRAM

Let us execute programs in different modes of programming.

INTERACTIVE MODE PROGRAMMING

Invoking the interpreter without passing a script file as a parameter brings up the following prompt –

```
$ Python
Python2.4.3 (#1, Nov112010, 13:34:43)
[GCC 4.1.220080704(RedHat4.1.2-48)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Type the following text at the Python prompt and press the Enter –

```
>>>print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");** . However in Python version 2.4.3, this produces the following result –

```
Hello, Python!
```

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension **.py**. Type the following source code in a test.py file –

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows –

```
$ Python test.py
```

This produces the following result –

```
Hello, Python!
```

FLASK FRAMEWORK:

Flask is a web application framework written in Python. Armin Rancher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

Http protocol is the foundation of data communication in World Wide Web. Different methods of data retrieval from specified URL are defined in this protocol.

The following table summarizes different http methods –

S.No	Methods & Description
1	GET Sends data in unencrypted form to the server. Most common method.
2	HEAD Same as GET, but without response body
3	POST Used to send HTML form data to server. Data received by POST method is not cached by server.
4	PUT Replaces all current representations of the target resource with the uploaded content.
5	DELETE Removes all current representations of the target resource given by a URL

By default, the Flask route responds to the **GET** requests. However, this preference can be altered by providing methods argument to **route ()** decorator.

In order to demonstrate the use of **POST** method in URL routing, first let us create an HTML form and use the **POST** method to send form data to a URL.

Save the following script as login.html

```
<Html>
```

```
<Body>

<formaction="http://localhost:5000/login"method="post">

<p>Enter Name:</p>

<p><inputtype="text"name="nm"/></p>

<p><inputtype="submit"value="submit"/></p>

</form>

</body>

</html>
```

Now enter the following script in Python shell.

```
from flask import Flask, redirect, url_for, request

app=Flask(__name__)

@app.route('/success/<name>')

def success(name):

return'welcome %s'% name

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

def login():

if request.method=='POST':

user=request.form['nm']

return redirect(url_for('success',name= user))

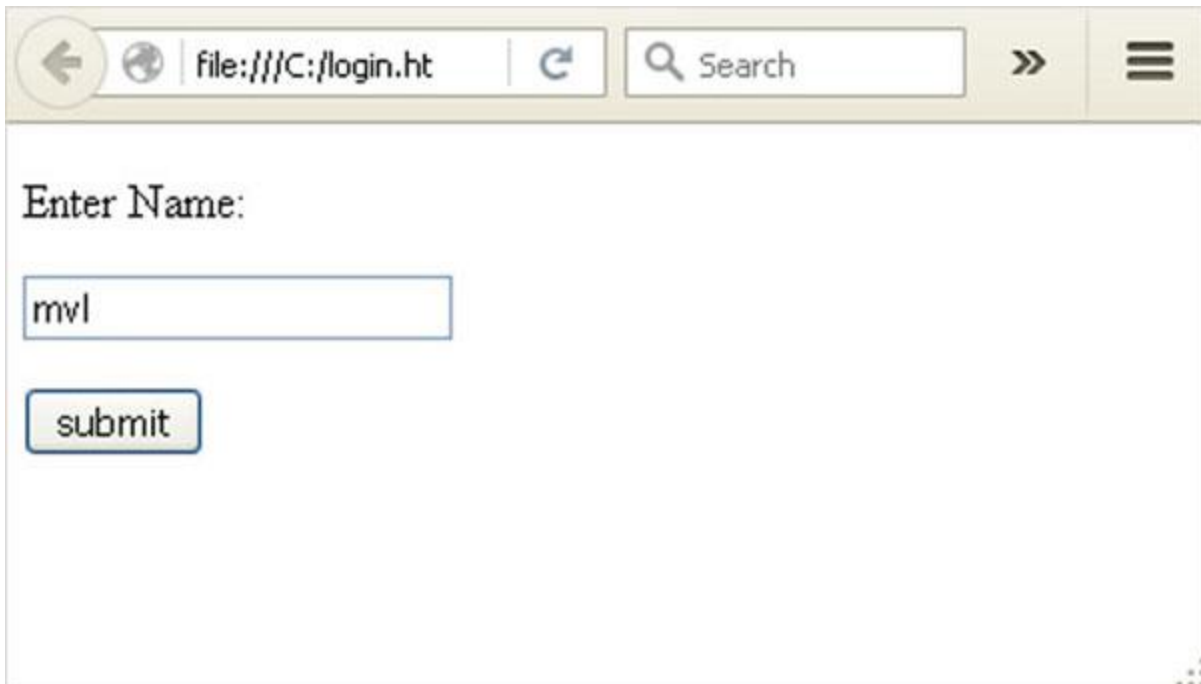
else:

user=request.args.get('nm')

return redirect(url_for('success',name= user))
```

```
if __name__ == '__main__':  
    app.run(debug = True)
```

After the development server starts running, open **login.html** in the browser, enter name in the text field and click **Submit**.



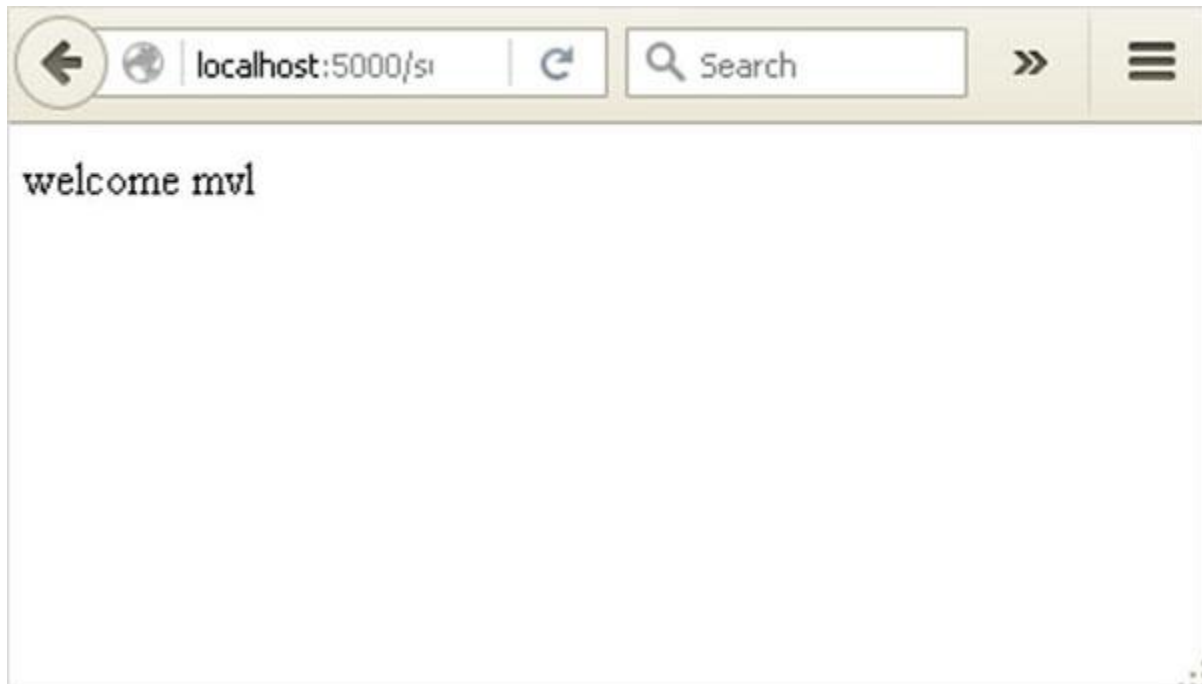
The screenshot shows a web browser window. The address bar contains the file path 'file:///C:/login.ht'. The main content area displays a form with the label 'Enter Name:'. Below the label is a text input field with the value 'mvl'. Underneath the input field is a button labeled 'submit'.

Form data is POSTed to the URL in action clause of form tag.

http://localhost/login is mapped to the **login()** function. Since the server has received data by **POST** method, value of 'nm' parameter obtained from the form data is obtained by –

```
user = request.form['nm']
```

It is passed to **'/success'** URL as variable part. The browser displays a **welcome** message in the window.



Change the method parameter to **'GET'** in **login.html** and open it again in the browser. The data received on server is by the **GET** method. The value of 'nm' parameter is now obtained by –

```
User = request.args.get('nm')
```

Here, **args** is dictionary object containing a list of pairs of form parameter and its corresponding value. The value corresponding to 'nm' parameter is passed on to '/success' URL as before.

CHAPTER 5

PERFORMANCE ANALYSIS, DISCUSSION AND RESULT

5.1 PREPROCESSING TRAINING DATASET

Dataset	Training Dataset		
	<i>Normal</i>	<i>Pneumonia</i>	<i>COVID-19</i>
number	7966	8620	128

5.2 PREPROCESSING TESTING DATASET

Dataset	Testing Dataset		
	<i>Normal</i>	<i>Pneumonia</i>	<i>COVID-19</i>
number	885	956	12

5.3 PURPOSE:

I wanted take this opportunity and apply one of the most famous survival analysis methods, called Cox Regression on this dataset. I am sure many of you are familiar with this one. We are dealing with right censored data, (left censored as well, but we could make the assumption that time 0 is the confirmed date), so survival analysis seemed to me really applicable in this case. We will build and evaluate the predictive power of our model, as well as extract some key results I would like to thank all involved into gathering this dataset. Additionally the author of the Lifelines python package, for implementing all sorts of Survival Analysis algorithms.

5.4 BUILDING THE DATASET:

patient_id	age_decade	is_male	died	recovered	survived
1000000001	5.0	1	0	1	13
1000000002	3.0	1	0	1	32
1000000003	5.0	1	0	1	20
1000000004	2.0	1	0	1	16
1000000005	2.0	0	0	1	24

5.5 SURVIVAL ANALYSIS - COX REGRESSION

I would not like to bore you with the details of survival analysis. These concepts are much better explained in the original [lifelines](#) documentation. The only thing that I want to describe is how I handle the events:

- died : If died = 1 the patient is deceased
- recovered : If recovered = 1 the patient has recovered
- Survived: Number of days that the patient has "survived".
- This is not to be confused with the recovery of a patient. It rather means how many days a patient survived while being observed.
- Survival in this case means not death and not recovery.
- Consequently, if the patient is not deceased or has he not recovered, the patient is censored, meaning we do not know the outcome yet.

At this point I would like to state that we are working within a "competing risks" setting. In effect, recovery from the virus is a competing risk of death and vice versa. Let's move on and split our dataset into train and test sets

5.6 HAZARD AND BASELINE HAZARD

In Survival Analysis, Hazard is a function of time

$h(t) = \lim_{\delta t \rightarrow 0} \frac{\Pr(t \leq T \leq t + \delta t | T > t)}{\delta t}$ 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

$$h(t|x)_{\text{hazard}} = b_0(t) \exp\left(\sum_{i=1}^n b_i(x_i - x_i^-)\right) \log\text{-partial hazard}$$

$$h(t|x)_{\text{hazard}} = b_0(t) \exp\left(\sum_{i=1}^n b_i(x_i - x_i^-)\right) \log\text{-partial hazard}$$

Where $b_0(t)$ is the baseline hazard and which is scaled up or down based on some regressors X

In this exercise I am using 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 I am using for predicting the hazard are the age decade of the patient and the gender.

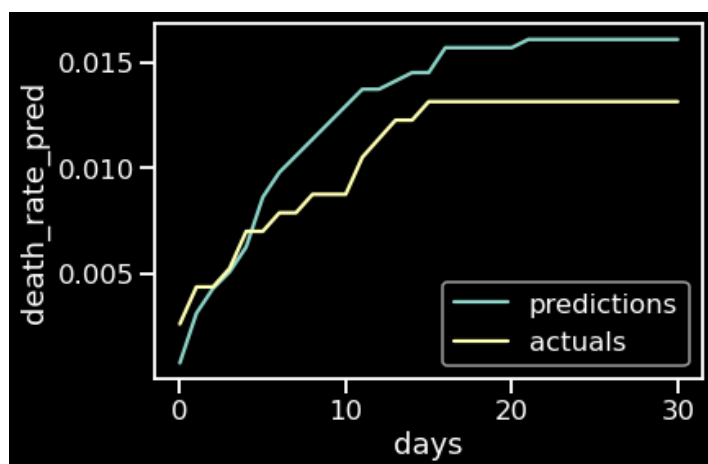
But since we are operating in a competing risks setting, we will have to fit 2 different models. The first one outputs the baseline hazard for the death event, while the second one will output the baseline hazard of the recovery event. An important distinction that we need to make is that we censor competing events at the time of the occurrence. Let's say for example that we are modeling 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 modeling 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. [An interesting read](#) for someone who wants to become more familiar with the topic.

Let's speak about the summary of the model after fitting is completed. Our concordance index is ~0.9, meaning that the features selected have good ranking power. If we revisit the Hazard function in Cox Regression, we will see that both coefficients of our futures are positive. In effect, this means that if we increase the age decade we increase the hazard of dying. Similarly men have a higher risk of dying than women.

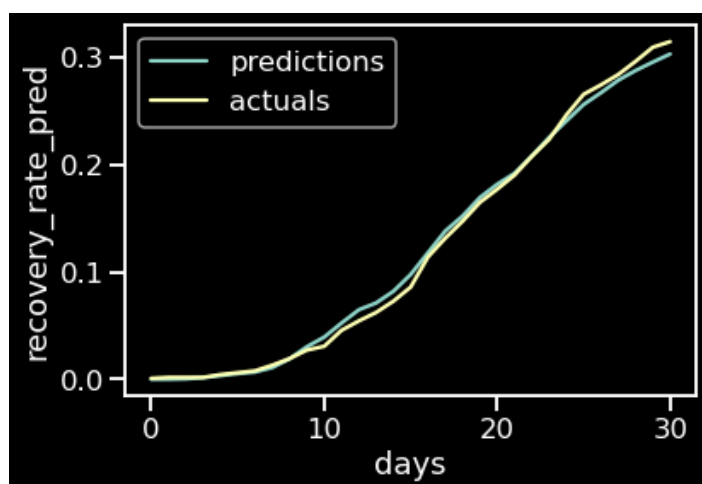
patient_id	baseline_hazard_death	baseline_hazard_recovery
0.0	0.000555	0.000000
1.0	0.001667	0.000000
2.0	0.000837	0.000594
3.0	0.000559	0.001785

4.0	0.000841	0.002981
-----	----------	----------

Because we will manually predict the hazard of each patient and because lifelines outputs the baseline hazard of a patient with average features (see [this issue](#) raised) we will construct the baseline hazards ourselves, by encoding a base case where all X's are set to 0.



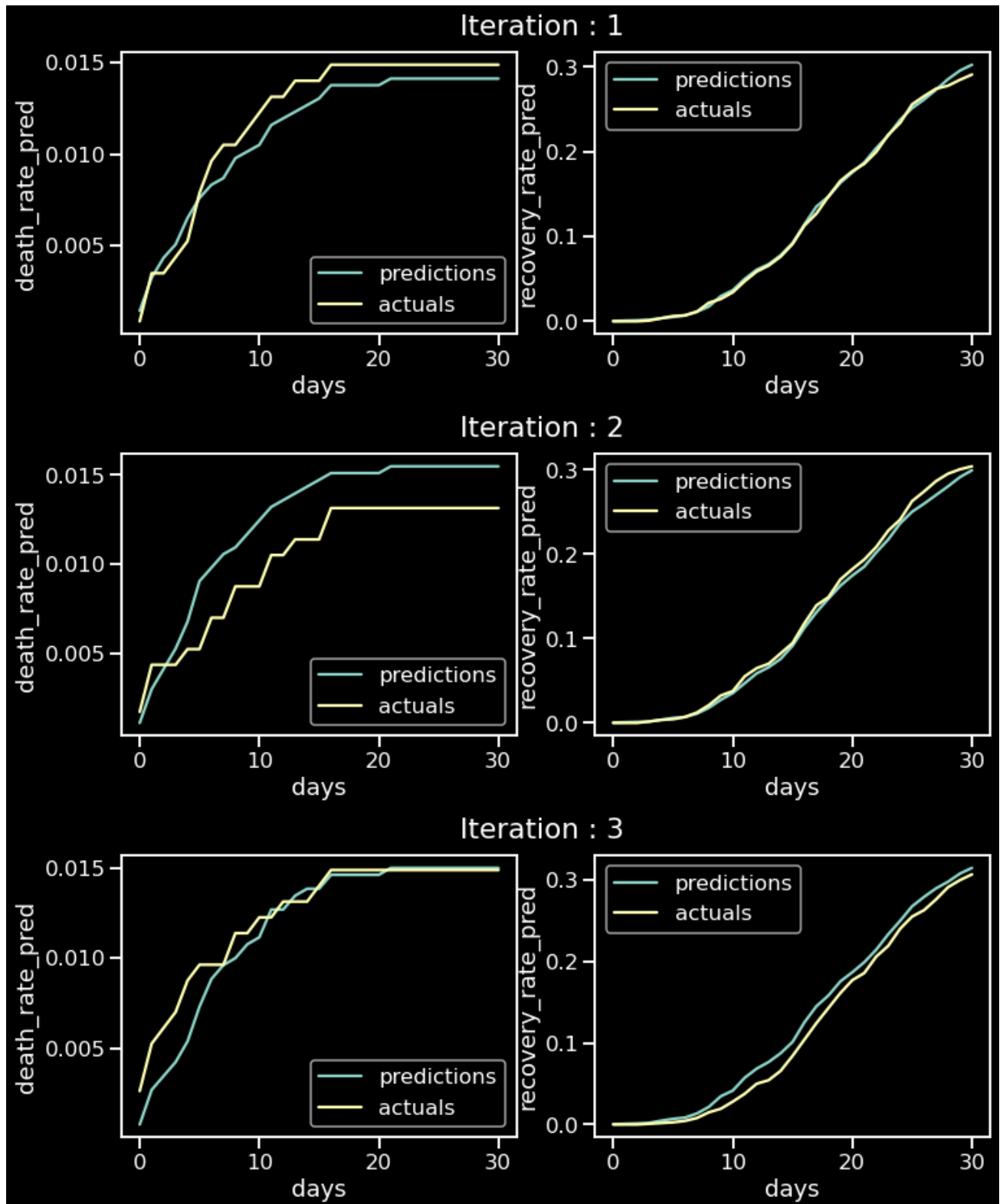
As you can see, the model seems to predict pretty well. Maybe the specific random fold of the data into train and test is such that favors prediction. What will follow is a "manual" cross validation, where we will iterate with different folds and average the errors. But before going into that let's observe the produced chart. According to the model, if 100 new patients enter the hospital, by 10 days ~1.5% of the patients will be deceased and by 30 days ~2%.

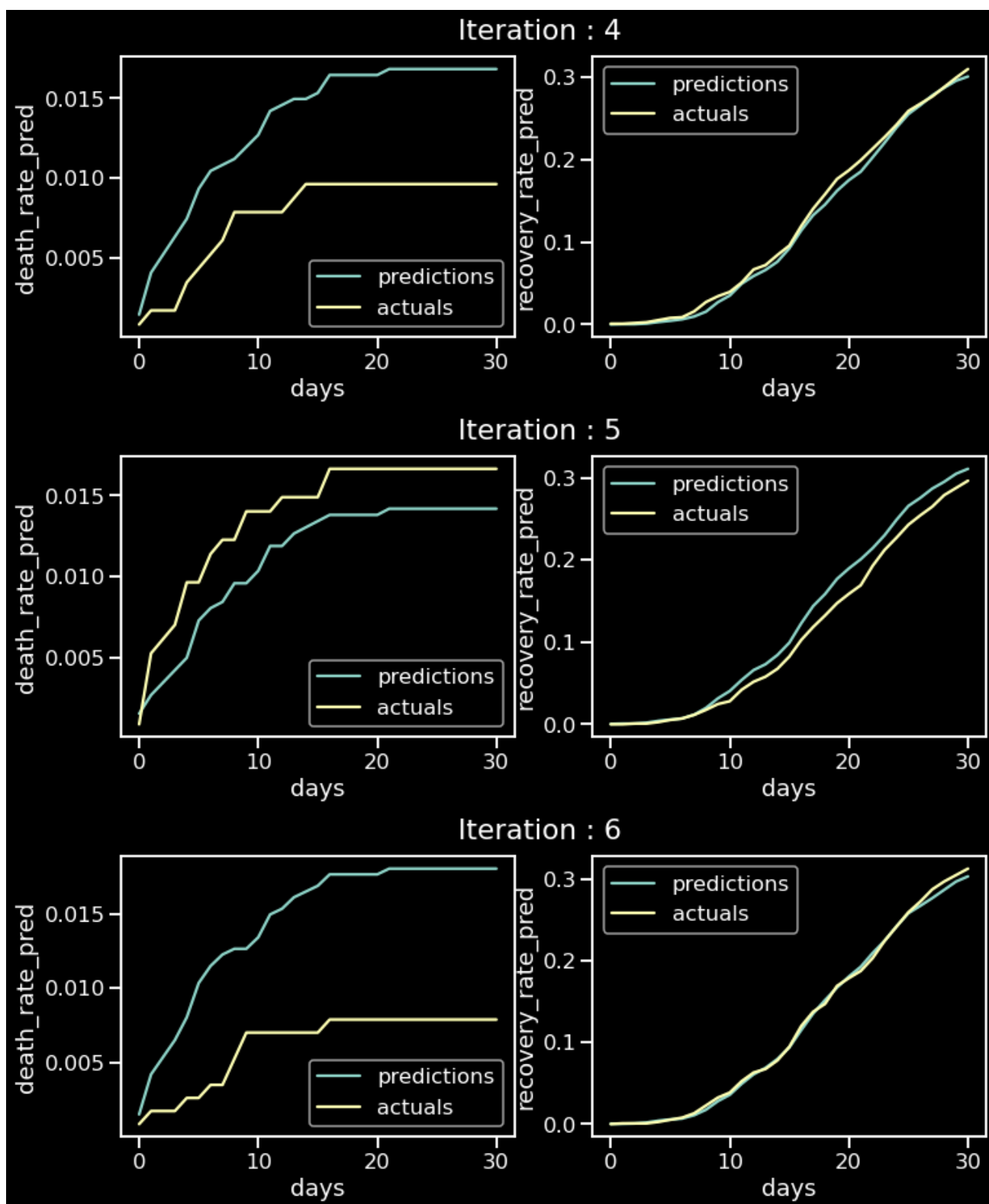


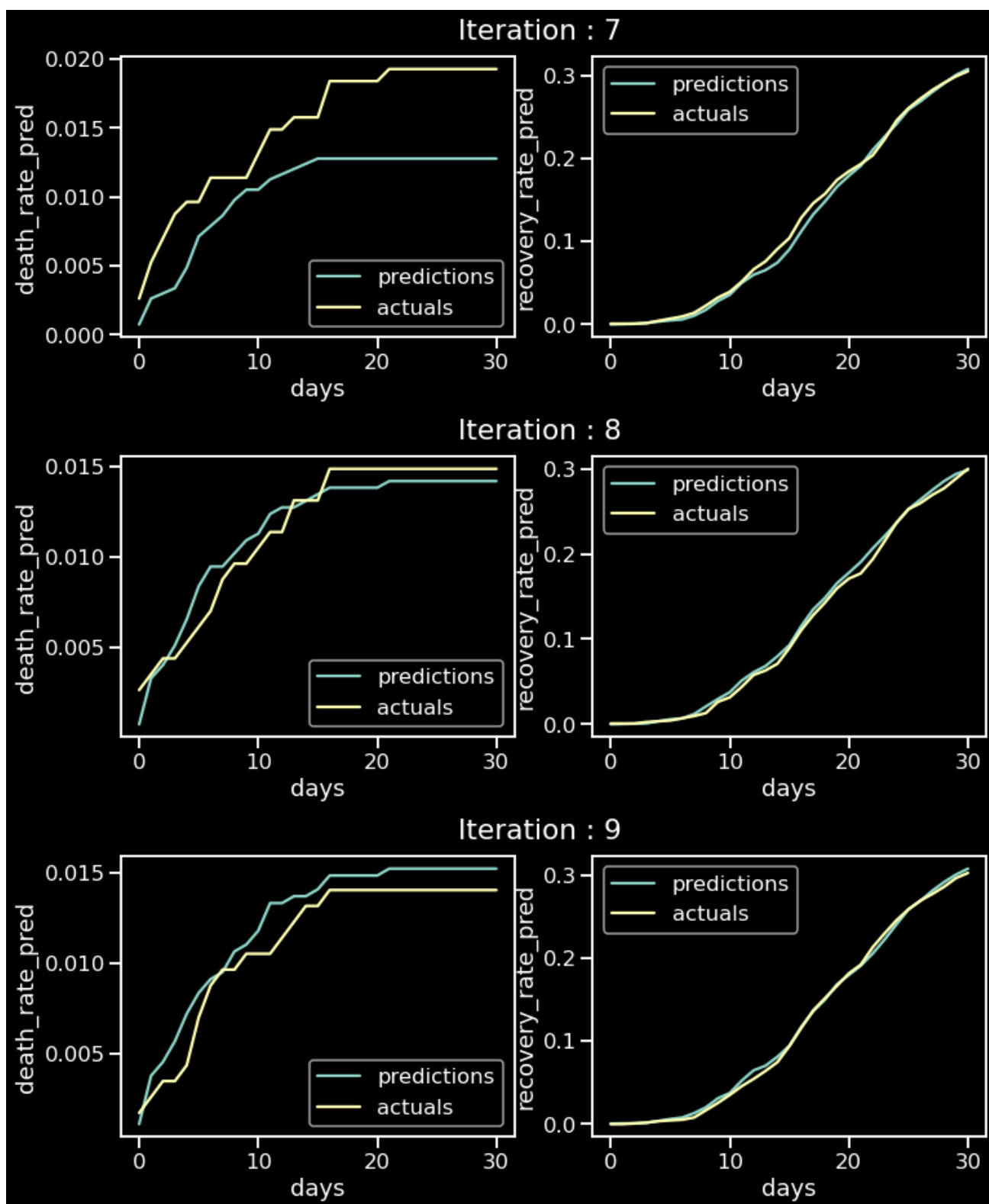
Similarly, the model predicts 40% recovery rate by the 30th after being diagnosed.

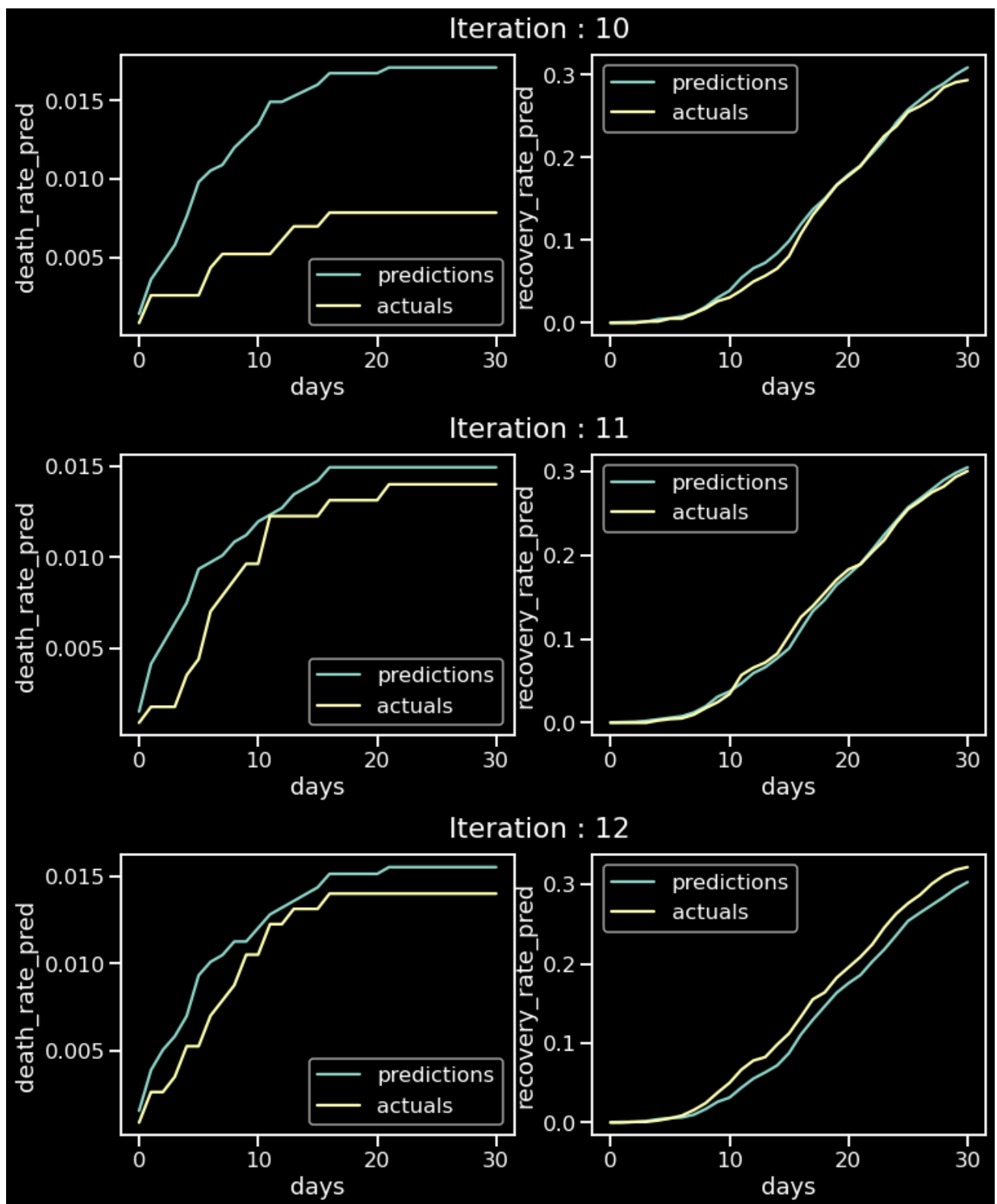
5.7 CROSS VALIDATION:

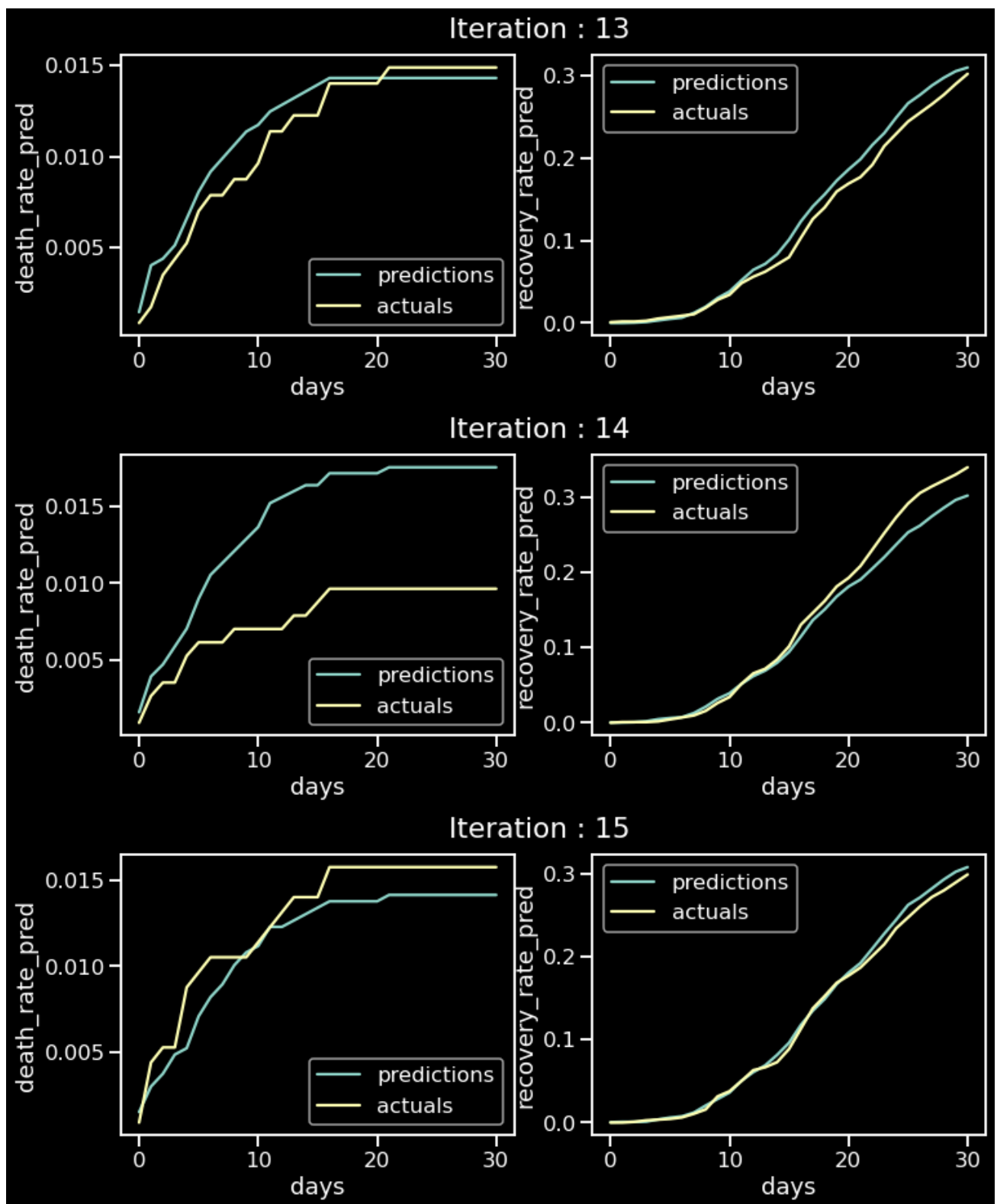
Since there is not a big amount of data so far, the split into train and test could make the results fluctuate a lot. For that reason I proceeded in coding a "manual" cross validation and I will take average error of each iteration as a final metric for the model.

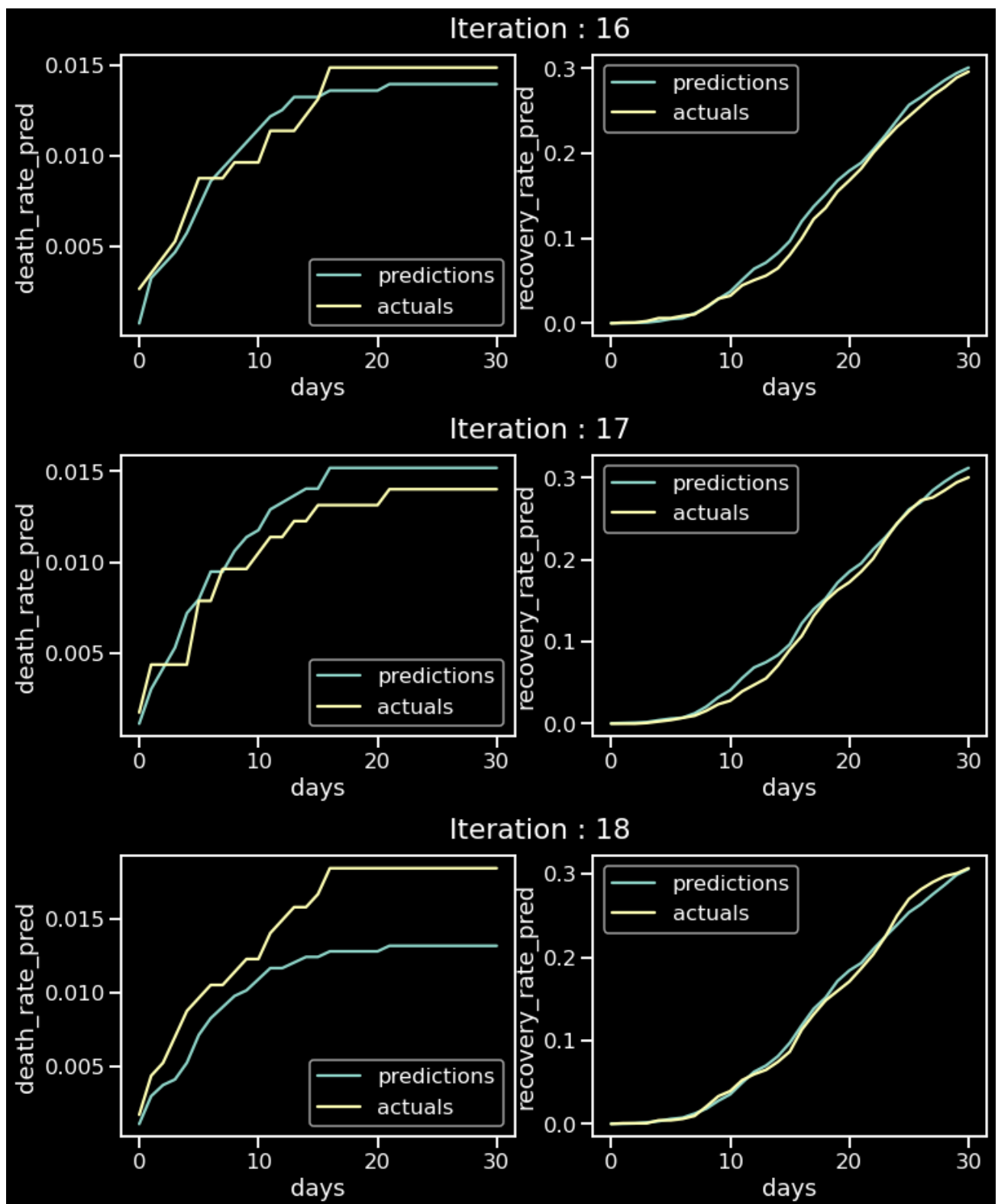


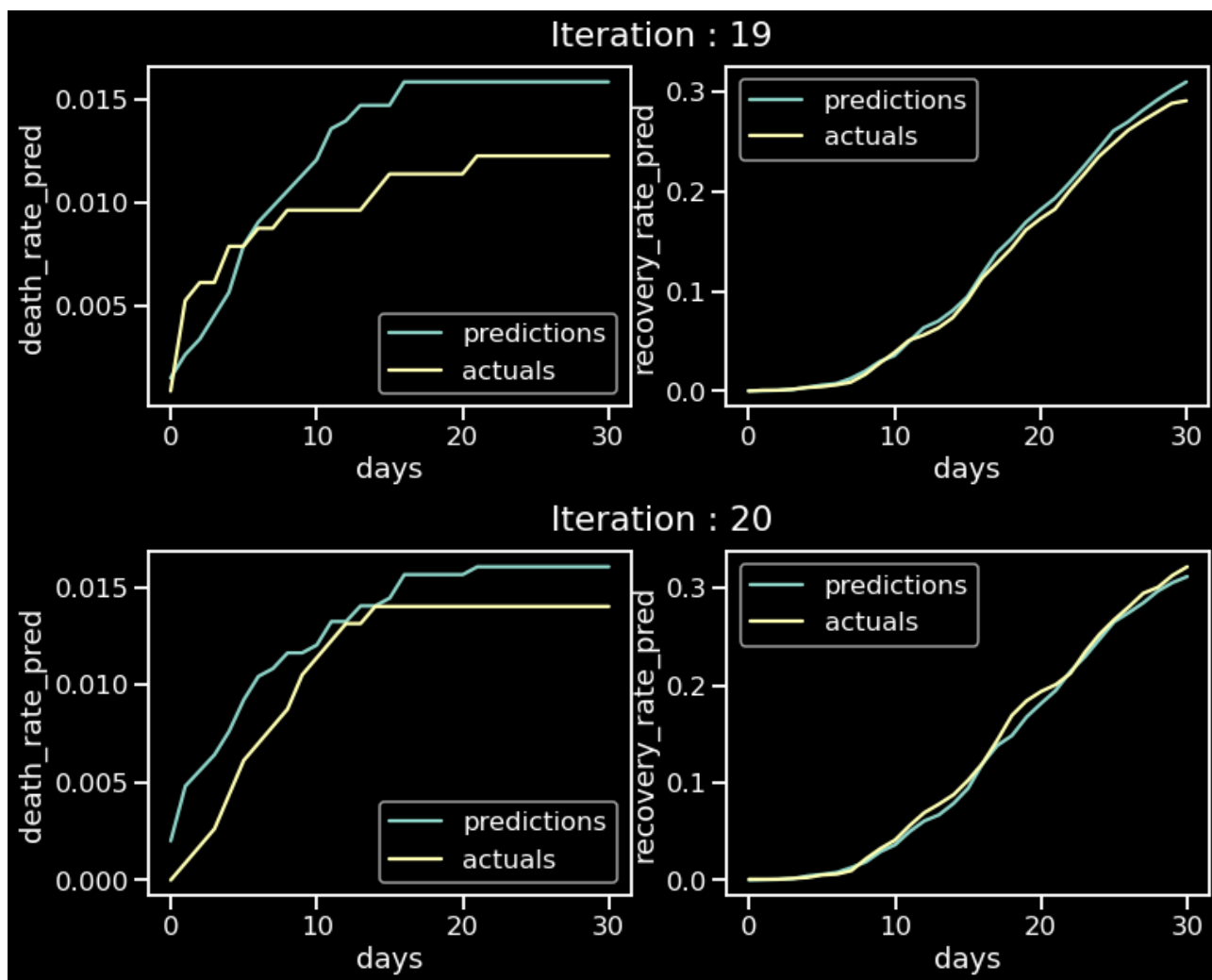






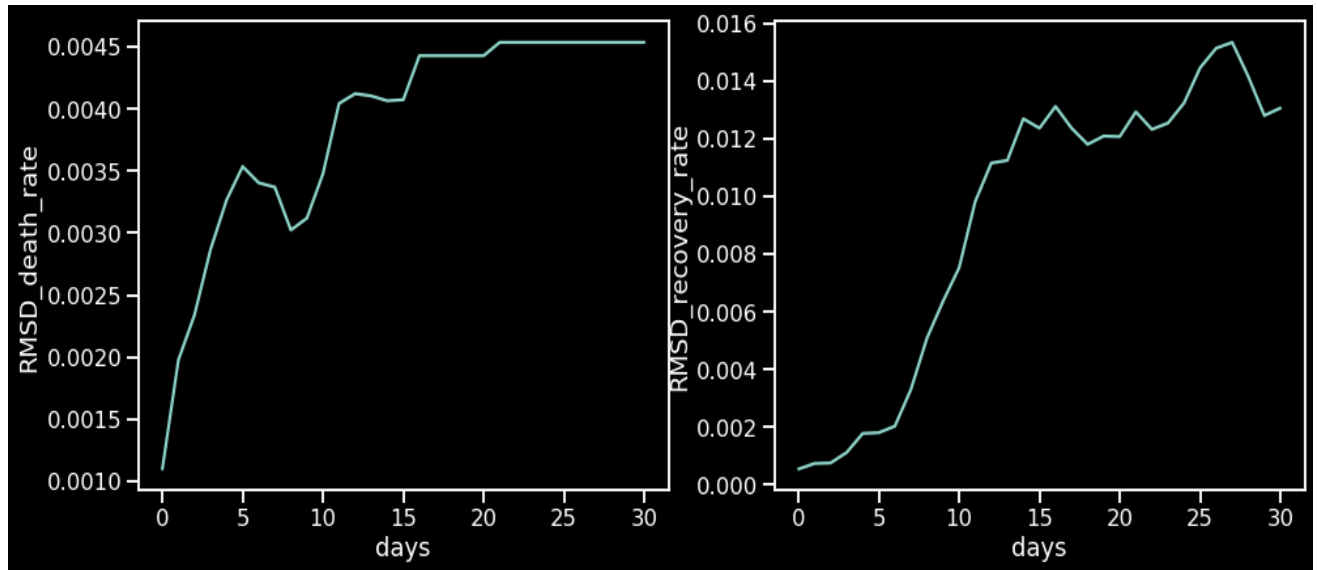






5.8 RESULT:

Our Root Mean Square Deviation when predicting death rates at 30 days is 0.6%. We need to be cautious about this number, since it depends on the magnitude of the number of deceased patients. It is not with certainty that if the death rate was higher, the RMSD of the death rate would still be 0.6%. Similarly our RMSD for the recovery rate is 10% at 30 days. For sure, the further we look into the future, the less accurate our results become.



CHAPTER 6

SUMMARY AND CONCLUSIONS

6.1 CONCLUSION

In this article we have studied the problem of locating persons in a given area, that should abide to social distancing measures such as those arising in the time of COVID-19 and similar viruses. We have argued that the p -dispersion problem can be used to efficiently model these situations. We devised a discretization scheme that was build on top of the decremented clustering method to get computationally attainable solutions, which worked very well in the computational study on the two artificial examples, especially for smaller values of p . We have investigated the effect of rounding of the dissimilarity matrix D on the computational effort and conclude that it is an indispensable part of the discretization scheme that has virtually no disadvantages in terms of the quality of the obtained solution. We have also seen the substantial increase in computational efforts for higher values of p , which can be contributed to the “dual degeneracy” of the clustering scheme. For future research, fast heuristics that run parallel to the decremented clustering scheme might further improve on the computational time and the size of the problems that are solvable by the presented method. Also, a mesh refinement scheme based on the current optimal placement (instead of the triangle size as presented here) could lead to improvements in the objective value.

7. REFERENCES

- [1] 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,
- [2] 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,
- [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] 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.
- [5] E. Erkut, "The discrete p-dispersion problem," *Eur. J. Oper. Res.*, vol. 46, no. 1, pp. 48–60, May 1990.
- [6]] M. J. Kubby, "Programming models for facility dispersion: The pdispersion and maxisum dispersion problems," *Geograph. Anal.*, vol. 19, no. 4, pp. 315–329, Sep. 2010.
- [7] E. Erkut and S. Neuman, "Analytical models for locating undesirable facilities," *Eur. J. Oper. Res.*, vol. 40, no. 3, pp. 275–291, Jun. 1989.
- [8]] 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.
- [9] D. Pisinger, "Upper bounds and exact algorithms for p-dispersion problems," *Comput. Oper. Res.*, vol. 33, no. 5, pp. 1380–1398, May 2006.
- [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] 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.

8. APPENDIX

8.1 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
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})
cummulative_baseline_hazard_death =
cph_death.predict_cumulative_hazard(base_case_death)
baseline_hazard_death =
cummulative_baseline_hazard_death.diff().fillna(cummulative_baseline_hazard_death)

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_reco
very)

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_death)

    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_reco
very)

    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['RMSE_death_rate'] = np.sqrt(errors.squared_death_diff / errors.iteration)
errors['RMSE_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.RMSE_death_rate, ax = ax1)
sns.lineplot(errors.days, errors.RMSE_recovery_rate, ax = ax2)

```