APPLICATION OF THE BLOCKCHAIN TECHNOLOGY WITHIN HEALTHCARE

MICHELLE LIN HWEI YEE

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Applied Mathematics with Computing

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

April 2021

DECLARATION

I hereby declare that this project report entitled "APPLICATION OF THE BLOCKCHAIN TECHNOLOGY WITHIN HEALTHCARE" is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : 

Name : Michelle Lin Hwei Yee

ID No. : 1701488

Date : 8 April 2021

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"APPLICATION OF THE BLOCKCHAIN TECHNOLOGY WITHIN HEALTHCARE"** was prepared by **MICHELLE LIN HWEI YEE** has met the required standard for submission in partial fulfillment of the requirements for the award of Bachelor of Science (Honours) Applied Mathematics with Computing at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Denis C K Wong

Date : 8 April 2021

ACKNOWLEDGEMENTS

I would like to thank everyone who had helped me throughout my project. I would like to express my very great appreciation to my research supervisor Dr Denis Wong Chee Keong for his valuable advice and led me to the right path. Without his patient guidance and persistent help, this final year project would not have been possible.

A special thank of mine goes to my lecturers, Dr Liew How Hui and Dr Ng Oon Ee, who had taught me the subjects Predictive Modelling and Artificial Intelligence. They had helped me to build my foundation knowledge on predictive models and contributed to the successful completion of this project.

Last but not least, I would also like to express my gratitude to my parents and friends for their support and encouragement.

ABSTRACT

Blockchain technology is an advanced technology that was initially used in the cryptocurrency field and now it has been extensively being implemented in various industries and fields. In this project, the potential of blockchain implementation in the healthcare industry is figured out and the area of study is focused on the COVID-19 related matters. In order to help with the situation of COVID-19 pandemic, a predictive model on the infected patient's severity level is constructed. The predictive model is used to forecast the severity level of a person who is infected with COVID-19 virus based on his various symptoms. In this project, a Logistic Regression statistical model is developed by using Python to construct the proposed model. Python is a popular programming language that is widely used in the area of artificial intelligence and it also has plenty of libraries that are used to build various predictive models. Besides that, a blockchain-based framework that focused on the Electronic Medical Record (EMR) is proposed. A broad study of journal papers on the blockchain framework is being carried out in order to have a basic understanding on the blockchain technology. The idea of the EMR framework with the blockchain-based application and database system is developed. Patients, hospital providers, healthcare authorities, financial parties, and the Certificate Authority are among the five groups of participants involved in the proposed framework. There are four proposed algorithms based on the actions of viewing and writing EMR, as well as another two additional actions aimed at improving the security of the blockchain framework. Finally, the constructed predictive model is implemented along with the proposed application, which connects all groups of participants with the blockchain-based EMR system.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

$h(x)$	prediction of y corresponds to the input x
x"	coordinate of training data x
y"	coordinate of training data y
$N(x)$	points which are close to input x
$P(Y = j \mid X = x)$	probability of getting particular y prediction based on inputs x
x	input x
j	particular output y
k	number of closest neighbors/points to input x
$P(Y = 1 \mid X_1=x_1, \ldots, X_p=x_p)$	probability of getting prediction Y
x	inputs x
β_i	parameter constant corresponds to x_i
$h(x)$	prediction of y corresponds to the probability obtained
p	optimal cut-off value
$P(Y = 1 \mid X = x)$	probability of getting prediction $Y = 1$ given input x
$P(Y = j \mid X = x)$	probability of getting prediction $Y = j$ given input x
x	inputs x
β_i	parameter constant corresponds to x_i
k	the level of output y
β_j	parameter constants corresponds to x_i at the level j
j	the particular level k for predicted y

ACP	Artificial system, Computational experiments and Parallel execution
AUC	Area Under the Curves
CA	Certificate Authority
CBNs	Consortium Blockchain Nodes
CMCO`	Conditional Movement Control Order
COVID-19	Coronavirus Disease 2019
CRUD	Create, Read, Update, Delete
DApp	decentralized application
DASH	Decentralized Application for Smart Healthcare
EDA	Exploratory Data Analysis
EMCO	Enhanced Movement Control Order
EMR	Electronic Medical Record
ERD	Entity Relationship Diagram
FHIR	Fast Healthcare Interoperability Resources
GA	Genetic Algorithm
IPFS	InterPlanetary File System
KNN	K-Nearest Neighbors
KSI	Keyless Signature Infrastructure
LR	Logistic Regression
MCO	Movement Control Order
PBFT	Practical Byzantine Tolerance Algorithm
PHSs	Parallel Healthcare System
PoW	Proof-of-Work
P2P	peer-to-peer
RMCO	Recovery Movement Control Order
ROC	Receive Operating Characteristics
WHO	World Health Organization
XDS	Cross-Enterprise Document Sharing

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 General Introduction

1.1.1 Background of Coronavirus Disease 2019 (COVID-19)

"COVID-19 is an infectious disease caused by a newly discovered strain of coronavirus, a type of virus known to cause respiratory infections in humans" (World Health Organization (WHO), 2021). It was first discovered in Wuhan, China in December 2019, then followed by the first reported cases from Thailand and United States of America (USA) in January 2020. It has been widely spread across the world leading WHO to declare the outbreak of it as global pandemic on 11 March 2020 (Susie and Aylin, 2020).

The COVID-19 disease is getting serious at the end of March as Italy reached its peak with 6,557 cases reported in a day while USA had a total of 82,404 confirmed cases, which is the highest number of confirmed cases in the world at that time. Within 4 months, the pandemic had caused the world to have more than 1 million infected cases and exceed 100,000 death cases. Due to the COVID-19 can be easily transmitted by human contact and social interactions are inevitable, the COVID-19 cases continue to grow dramatically. Several countries, including Italy, USA, Iran, Thailand and Malaysia, had announced their lockdown as a prevention to reduce the spread of COVID-19 (Susie and Aylin, 2020).

"Globally, as of 3:10pm CET, 16 March 2021, there have been 119,960,700 confirmed cases of COVID-19, including 2,656,822 deaths, reported to WHO" (World Health Organization (WHO), 2021). As of 16 March 2021, the country which had the most COVID-19 cases is the USA with the number of 26,155,892 cases, followed by Brazil and India, with the number of 11,483,370 and 11,409,831 cases respectively.

In Malaysia, the first reported COVID-19 cases was in January 2020 whereby a Wuhan family of a mother and two children travelled to Johor Bahru. In February 2020, the first COVID-19 infected Malaysian was reported, who had contact with China delegation when attended a meeting in Singapore. The COVID-19 cases were quite low in the beginning until the Tablighi

Jamaat religious gathering at Masjid Jamek, Sri Petaling sparked the first wave of infections in March 2020. Later on, the second wave was started around May 2020 due to the infections among migrant workers while third wave began in September 2020 which was caused by the Sabah state election.

Malaysia had hit the highest confirmed cases on 30th January 2021 with 5,728 confirmed cases in one day. To cope with the rapid growth of COVID-19 cases, Malaysia government had declared a lockdown known as Movement Control Order (MCO) on 18 March 2020 which lasted for 8 weeks. For the remaining weeks, different stages of MCO are introduced to control the situation of COVID-19 cases including Conditional MCO (CMCO), Recovery MCO (RMCO) and Enhanced MCO (EMCO).

The COVID-19 virus mainly spread through droplets of saliva and can be transmitted from human to human when there is a close contact with infected person. People who are infected have the common symptoms of dry-cough, fever and tiredness. New symptoms keep emerging such diarrhea, headache and loss taste and smell, and even evolved into asymptomatic. Young patients usually face mild symptoms and can be recovered easily. However for older patients, they are more likely to suffer from serious symptoms such as difficulties in breathing and need hospitalization for proper treatments.

Around December 2020, the first vaccine named Pfizer-BioNTech was successfully developed by USA and Germany with a 95% overall efficiency. As of March 2021, there have been several COVID-19 vaccines developed by different countries, including AstraZeneca, Sinovac, CanSinBIO and Sputnik V, with different technologies being implemented. “As of 15 March 2021, a total of 326,858,656 vaccines doses have been administered” (WHO, 2021).

1.1.2 Background of Blockchain Technology

Blockchain technology was first being introduced in 2009 and the application which implemented this technology is known as Bitcoin. Zyskind, Nathan and Pentland (2015) have mentioned that Bitcoin is a system that allows users to transfer currency (bitcoins) securely without a centralized regulator by using a publicly verifiable open ledger. In other words, the receiver is able to receive the money directly from the sender without the third-party such as bank to

process and validate the transaction. Frankenfield (2020) has stated that, it follows the ideas set out in a whitepaper by the mysterious and pseudonymous Satoshi Nakamoto.

Blockchain is a distributed ledger technology that works on the peer-to-peer (P2P) network, where the data are immutable once they are written and being stored as part of the blockchain. Each of the data is stored as a block and it is connected to another block by using hash function as the chain that links both of them together, which correspond to the name itself, blockchain.

Besides that, decentralization is another property of blockchain technology which makes it special and stands out. The data stored in blockchain is controlled by every of the participants in the system, where they are able to access all the data and add in new data. From here, it gives the blockchain a transparent property. Blockchain technology also grabs the attention of people as the data are being stored securely. Flynt (2016) have stated that, this honor goes to the cryptography that protects the data from being tampered with and the level of security that protects blockchain contents is currently considered nearly impossible to break.

1.2 Importance of the Study

With the special properties of blockchain, it is not only beneficial in the cryptocurrency area but it begins to be applied to various fields, including medicine, economics, Internet of things, software engineering and so on (Li, et la., 2017). There are some of the research papers have proposed the implementation of blockchain technology in the healthcare industry such as pharmaceutical supply chain and health insurance claims.

Hölbl, et la. (2018) have stated the findings in his research shows that blockchain technology research in healthcare is increasing and it is mostly used for data sharing, managing health records and access control. “In recent years, blockchain technologies have been applied in Electronic Medical Records (EMR) systems to provide control, supervision, accessibility, auditability, and interoperability over large scale data management frameworks using a comprehensive log” (Zubaydi, et la., 2019). The EMR is the electronic version of patients’ medical and treatment records, instead of using paper and pens to record down and save the details.

As COVID-19 is a new disease that was discovered in 2019, there are no specific treatments that can be referred to in order to treat the patients and the patients might not be fully recovered after their completion of treatment. In around August 2020, there is even a mutation of COVID-19 occurred in Malaysia that the COVID-19 virus tend to be 10 times more infectious than its original form. Research are needed to be carried out continuously on this COVID-19 disease in order to resolve it and flatten the curve. With the study of blockchain implementation onto COVID-19 patients' EMR, it would be beneficial to all of the doctors and researchers for them to discover more advanced COVID-19 treatments and vaccines to improve on the research progress.

1.3 Objectives

There are two objectives that need to be achieved throughout this project. The first objective is to utilize the data of COVID-19 and analyze them to build a model that would be useful to be shared in the blockchain system. Secondly, an application need to be proposed by implementing the blockchain technology into the healthcare industry, with the following features : (i) the severity level of COVID-19 patient model are able to be shared in the blockchain system, (ii) COVID-19 patients records are able to be stored in the blockchain system and shared among the healthcare entities in a more efficient and reliable way, (iii) COVID-19 patients have the control on the accessibility of their own information and records.

1.4 Problem Statements

In order to achieve the above two objectives, there are two problem statements which need to be tackled. The first problem statement would be "what are the suitable type of COVID-19 data to be collected in order to obtain an analysis result that would be beneficial in the medical or clinical research purpose?" whereas the second problem statement would be "how can the implementation of blockchain technology enable authorize parties such as the health ministry, healthcare centre, medical research centre and financial parties to keep track of COVID-19 data and patients' records in a more secure and reliable way?".

1.5 Expected Outcomes

From the outcome of analysis on COVID-19 data, a model on the severity level of COVID-19 patients is constructed. As the model is shared in the blockchain system, healthcare entities who are interested in the research of COVID-19 are able to utilize the model and give them an insight of it. This could help to contribute to the healthcare industry to conquer the COVID-19 pandemic. Besides that, medical and healthcare sectors could refer to this model and improve on their diagnosis results without missing out the subtle symptoms of infected patients. Financial parties are also able to make use of the predictive model and provide appropriate levels of financial support to COVID-19 patients.

On the application that integrate with the blockchain technology, the patient records are able to be stored into the blockchain in a secure manner. The doctors are able to keep track the patient condition by looking at their medical records. The patient's record were kept in its original form and they were verified on their authenticity.

For the clinical and research purpose, the patients' records are able to be shared among the healthcare authorities in a more efficient way. The healthcare authorities are able to view the patient's record as long as they are part of the blockchain system. The blockchain system is a distributed ledger enabling participants who connected to it to have a copy of data. This has helped the sharing of data to be more convenient without the need of a hospital's database admin to approve the sharing action. As the COVID-19 is not easily wiped out and the COVID-19 vaccines are successively being developed, exchanging the patient records would be beneficial for research purpose.

With the advanced version of blockchain technology, the patients could have their own control on who can access their personal information and records. Patients do not need to worry on their data is being leaked out or used for any illegal purpose.

1.6 Planning

During the stage of Project I, the registration of this project title "Application of the Blockchain Technology within Healthcare" is done at week 1 by getting the approval of supervisor. To start off, background study on the COVID-19 and blockchain are carried out to have a general understanding on them and extract the important knowledge for the information collection. Besides that, it is necessary to do literature review in order to learn about the most recent research findings and to gain thorough understanding of the project title. The studied academic papers are mainly related with the implementation of blockchain technology on the healthcare system.

In the middle stage of Project I, project proposal is submitted and a mock presentation is conducted based on the first 7 weeks of studies. The studies are continued until the end of Project I to submit an interim report and conduct a presentation to the supervisor and facilitator. Throughout this 14 weeks, there are a total of 7 biweekly reports submitted to supervisor for tracking on the project progress.

Task	Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Registration of Project Title														
Background Study														
Collect Information														
Study on Literature Review														
Submit Biweekly Reports														
Mock Presentation														
Submit Proposal														
Write Interim Report														
Submit Interim Report														
Project I Presentation														

Figure 1.1: Gantt Chart for Project I

After all the studies and preparation were done on Project I, Project II is focused on the development part to meet the objectives of the project. In the first 2 weeks of Project II, the suitable COVID-19 dataset is collected on the Kaggle website. The “COVID-19 Symptoms Checker” dataset is chosen as it contains the symptom features which can achieve the goal of this project. It is then used for building the models to predict the COVID-19 patient’s severity level. At the same time, the prototype of proposed application is developed. All the results and justifications are written into the report.

At the middle stage of Project II, a mid-semester monitoring report is submitted to the Final Year Project coordinator. A poster is made to summarize the content of final report and it is submitted at week 10 along with a pre-recorded presentation video. The final report is completed and submitted at week 12, followed by a presentation of it at week 13.

Task	Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Study on the Collected Data														
Build the Predictive Model on COVID-19 Patient’s Severity Level														
Develop the Prototype of Website														
Prepare Final Report														
Submit Mid-semester Monitoring Report														
Prepare FYP Poster														
Submit FYP Poster Presentation Video														
Submit Final Report														
Project II Presentation														

Figure 1.2: Gantt Chart for Project II

CHAPTER 2

LITERATURE REVIEW

2.1 Overview in Blockchain

Blockchain technology acts like the database system for the data to be stored, in the form that data acts as a block and it is appended to the end of the blockchain in the chronological order. In contrast with the traditional database which focuses on the operation of Create, Read, Update and Delete (CRUD), the data with the implementation of blockchain technology are only able to carry out the read and write operation.

When a block is being added into the blockchain, the timestamp and unique hash automatically generated and stored as part of it. The block also contains the hash of its previous block and its hash is also stored into the block that appends to it. The unique hash resembles its fingerprint and any slightest changes on the block would generate new hash for the block itself. This would causes the remaining blocks behind it to become invalid as the hash that points to the previous block "disappear". Thus, the data are unlikely to be overwritten or deleted, giving the blockchain of immutable property. The overview of blockchain is shown in Figure 2.1 (Angraal, Krumholz and Schulz, 2017).

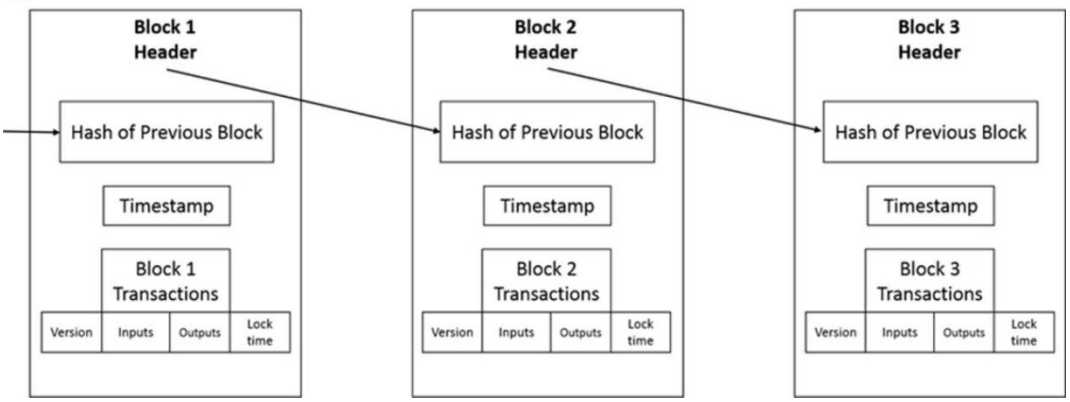


Figure 2.1: The Formation of Blockchain (Angraal, Krumholz and Schulz, 2017)

As blockchain technology works on the P2P network, every computer that connects to this blockchain system is able to access the data and they are known as the node. “The P2P systems that involve the interaction between contractual partners instead of indirect interaction through middleman helps in reducing the processing time and costs” (Drescher, 2017). In order for the block to be validated and verified, it must go through the consensus protocol whereby every of the nodes agree on it. This process is known as mining while the participants are the miners who carry out this process. The overview of mining process is shown in Figure 2.2 (Zubaydi, et la., 2019). There are different types of consensus algorithms are being introduced while the Proof-of-Work (PoW) is one of the widely-used consensus algorithm. Watanabe, et la. (2015) have mentioned that the PoW process resembles solving an uneasy puzzle in order to ensure the block reliability.

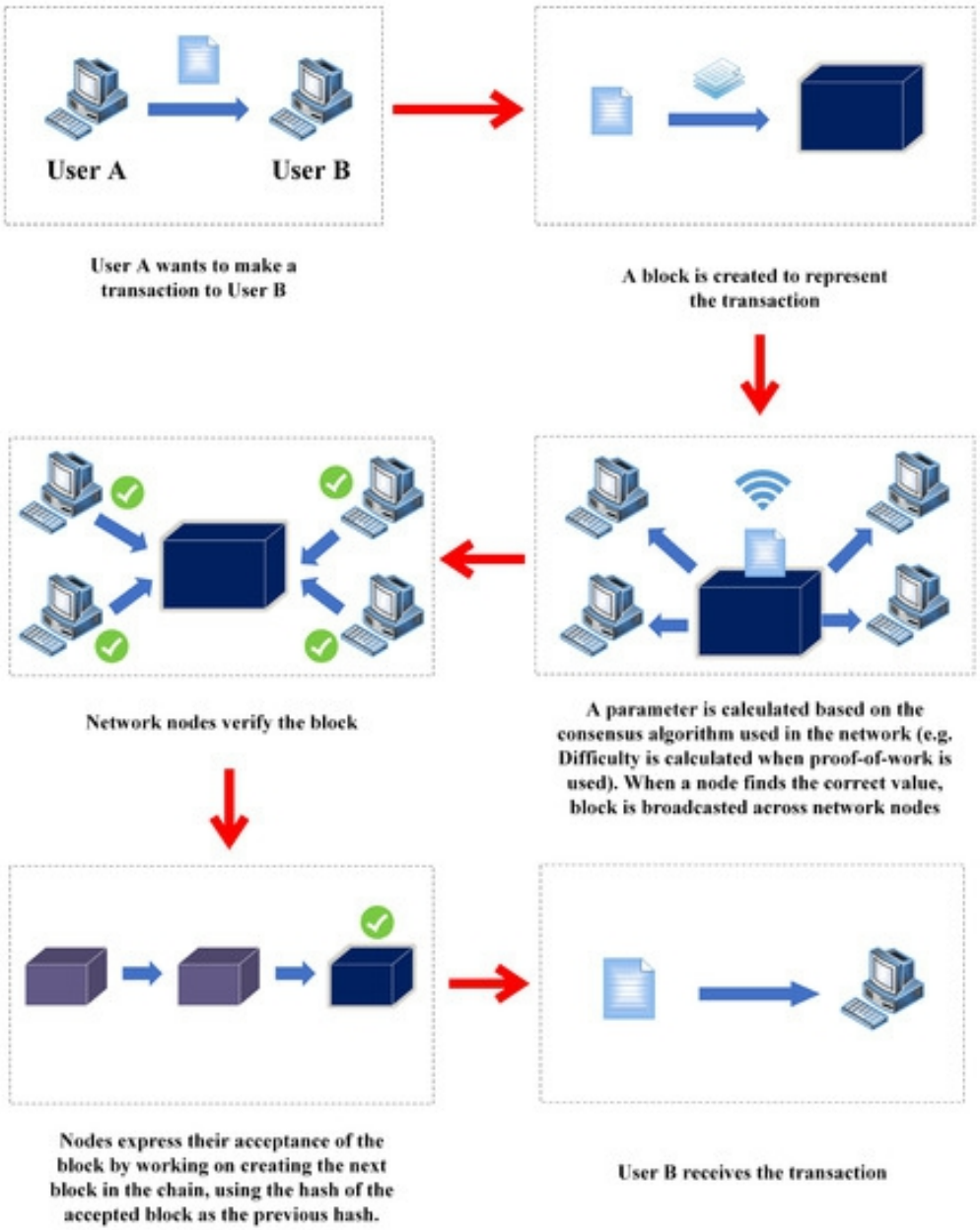


Figure 2.2: The Overview of Mining Process (Zubaydi, et la., 2019)

2.2 Types of Blockchain

The blockchain technology has been constantly being modified and improved so that it can be implemented in different fields and industries. The first version of blockchain is based on the currency concept as it is first implemented in the cryptocurrency application, Bitcoin. Later on, blockchain 2.0 is being introduced with the concept of smart contracts. The latest blockchain version has comes to 3.0, where it can be implemented on an application and it is known as decentralized application (DApp).

In general, there are three types of blockchain that differ from their level on data accessibility and data management. The first type is public permissionless where the public is able to access, view and mine the data. When there are only a certain group of people are allowed to access and mine the data, it narrows down the first type of blockchain to public permissioned or it is known as consortium. The third type is private blockchain which only the chosen miners are able to join the network.

2.3 Implementation of Blockchain on the Healthcare Industry

With the implementation of blockchain onto the EMR, the patients' data and records are protected from the malicious activities such as hacking. Any modification done on the data is not allowed and the consensus protocol prevents abnormal data to be validated and added into the blockchain. This improves the data integrity and keeps the data in a reliable way.

"What is more, the use of cryptographic algorithms to encrypt the data stored on the blockchain ensures that only the users who have legitimate permissions to access the data can decrypt them, thereby improving the data security and privacy" (Agbo, Mahmoud and Eklund, 2019). Furthermore, the distributed ledger technology also enables every computer (node) in the network to have a copy of the original blockchain. This simplifies the sharing of data in a secure way without transferring a large number of data between the healthcare entities.

There are some of the blockchain technologies have been implemented in the healthcare industry and successfully developed healthcare applications such as MedRec. Angraal, Krumholz and Schulz (2017) have stated that MedRec is a platform that offers a decentralized approach to manage permissions, authorization, and data sharing between healthcare systems. In 2015, the first version of MedRec was developed on the Ethereum blockchain by utilizing its smart contract system. Since EMR are designed for the use of only one healthcare organization, the MedRec was created to improve the usage of EMR by gathering patient data across multiple healthcare organizations into single access. It provides a convenient way to access data and save time on collecting patient data from scattered places. The MedRec system is keep improving and working on its version 2.0 currently.

"Another famous example is Guardtime, a company providing a blockchain-based system in Estonia to secure 1 million health records" (Kuo, Kim and Ohno-Machado, 2017) . Guardtime is founded by Mike Gault and it is a cyber-security provider that makes use of blockchain technology and offers services. It has invented the Keyless Signature Infrastructure (KSI), integrating the blockchain technology and Oracle database to achieve data authenticity without third parties. Estonia has taken the first move to be the first country that manages medical records using blockchain technology through its collaboration with Guardtime.

2.4 Related Work

With the emerging trend of blockchain technology and its unleashed potential in healthcare system, the research done on this area has been increased exponentially in recent years. Most of the proposed framework or prototype are built on the consortium blockchain. As the patient records and EMR are highly confidential and sensitive, the consortium blockchain that limits the number of miners and, at the same time, able to make the system to be transparent to everyone, consortium would be the suitable type to be chosen for the implementation of blockchain into healthcare system.

Wang, et la., (2018) have proposed a framework of parallel healthcare systems (PHSs) with the implementation of blockchain and added on the ACP approach feature. The ACP approach feature have combined the artificial system, computational experiments and parallel execution, which makes their proposed framework to have a higher accuracy rate on diagnosis and improve the patients' treatments. They have also constructed a prototype named parallel gout diagnosis and treatment system in their research paper.

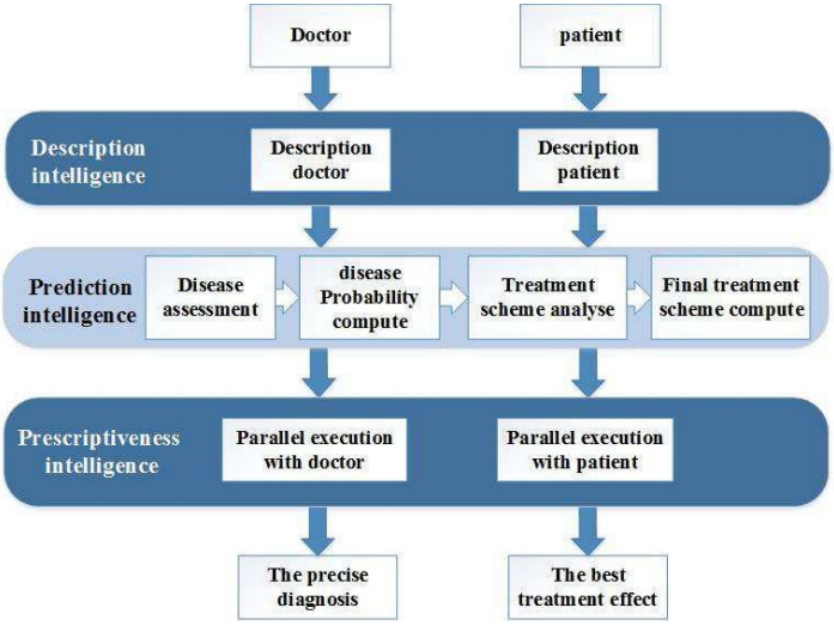


Figure 2.3: Parallel Gout Diagnosis and Treatment System (Wang, et la., 2018)

Usman and Qamar (2020) have proposed another blockchain framework that focused on improving the patient record management system. Their main purpose is to improve the efficiency of managing and sharing EMR. With that purpose, they have implemented their proposed prototype of Electronic Medical Records Management System with one of the popular permissioned blockchain platform named Hyperledger that uses Practical Byzantine Tolerance Algorithm (PBFT) .

Lee, Kim and Kim (2019) have proposed SHAREChain, a data sharing framework with the implementation of Blockchain-registry's data integrity, Fast Healthcare Interoperability Resources (FHIR) and Cross-Enterprise Document Sharing (XDS). Their main focus is on improving the data reliability and overcoming the interoperability issues on current healthcare system.

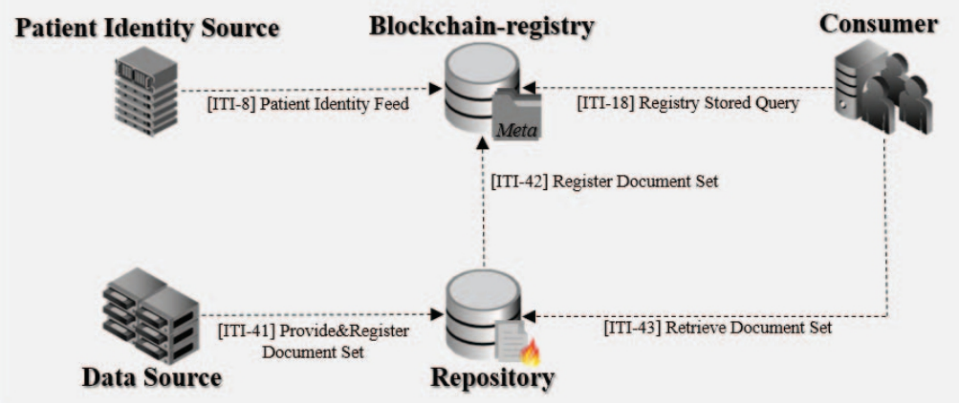


Figure 2.4: Actor Diagram using Blockchain-register and FHIR to XDS Framework (Lee, Kim and Kim, 2019)

Besides, there is another decentralized data management system named HealChain presented by Ni, et la. (2019) with their focus on mobile healthcare. They made use of the consortium blockchain nodes (CBNs) and combined it with the InterPlanetary File System (IPFS). They also presented the Genetic Algorithm (GA) as a solution to maximize the mining process in CBNs and improve its optimum performance.

Furthermore, Christ, et la. (2019) studied the usage of blockchain into healthcare system to look for the potential in solving the current issues faced by the Indonesia healthcare system and thus improve the performance of it. They have reviewed the decentralized application for smart healthcare (DASH) model proposed by Kurniadi and Astuti that helps in conquering the problems in EMR. They have also found out MedRec provides the most suitable architecture to implement into the Indonesia healthcare system, thus they modified it to fulfill the local conditions.

Additionally, Vora. Et la. (2018) presented a blockchain-based framework that enhances the EMR performance in terms of storage and maintenance by giving out the idea of having five different kinds of contract to be processed in the blockchain. Tanwar, Parekh and Evans (2020) have also proposed a similar framework to work on the EMR and did a performance evaluation on their own proposed blockchain-based framework that uses the hyperledger fabric.

From the above proposed blockchain-based systems in healthcare industry, six of them are focused on patient data management and the remaining one has the purpose of improving diagnosis results. Majority of the proposed systems are inspired by the EMR to further enhance its usage with the implementation of blockchain technology. Various methodologies have been used across the mentioned journals, including the detailed architecture, mobile application proposal and performance evaluation of the proposed system.

Literature review has shown that there are much more potential of the blockchain being implemented into the healthcare system, thus this project is meant to contribute to the healthcare field by proposing a better blockchain-based system. In this project, a framework that could help in the doctors' diagnosis and patient data management is proposed, not to mention the website prototype which illustrates its concept. In contrast with the proposed systems mentioned in the above articles that store only the patient metadata into database, the blockchain-based database system is introduced in this project.

The comparison between seven of the proposed blockchain frameworks have been summarized in Table 2.1. In general, all of the proposed frameworks have improved on the data reliability as blockchain technology able to secure the data using cryptography and enhance the interoperability for the convenience of data sharing.

Journal Title	Year	Authors	Advantages	Disadvantages
Blockchain-Powered Parallel Healthcare System Based on the ACP Approach	2018	Wang, et la.	Forecast the patient disease and guide doctors in improving the diagnosis accuracy	Not mentioned
BHEEM: A Blockchain-based Framework for Securing Electronic Health Records	2018	Vora. Et la.	Further improve the data security due to the various proposed contracts	Not mentioned
SHAREChain : Healthcare data sharing framework using Blockchain-registry and FHIR	2019	Lee, Kim and Kim	Improve the reliability and interoperability of healthcare data	- stored data cannot be removed - it is not a patient-centred system
HealChain : A Decentralized Data Management System for Mobile Healthcare Using Consortium Blockchain	2019	Ni, et la.	Enhance the authenticity of data and improve the convenience on uploading data from mobile	Not mentioned
Exploring Blockchain in Healthcare Industry	2019	Christ, et la.	Improve the performance of healthcare system specifically of Indonesia	might not implementable in country that have immature infrastructure
Secure Electronic Medical Record Storage and Sharing Using Blockchain Technology	2020	Usman and Qamar	Improve the efficiency and security of managing and sharing of EMR	Not mentioned
Blockchain-based electronic healthcare record system for healthcare 4.0 applications	2020	Tanwar, Parekh and Evans	Proposed a framework and further improved it through optimization	Not mentioned

Table 2.1: Summary on the Related Work

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Research Flowchart

The primary goals of this project are to develop a predictive model on the severity level of COVID-19 patients and to propose a blockchain-based framework within healthcare system. Each goal is broken down into three stages to provide a step-by-step guideline on achieving it. To achieve the first goal, the COVID-19 data is collected from Kaggle and being analysed to determine its characteristic and potential usage. Following that, the COVID-19 data is used to build a predictive model using Python programming language. To achieve the second goal, academic articles on the topic related to blockchain technology within healthcare are extensively studied. Later on, ideas on the blockchain-based framework and algorithm are developed to meet the project objectives. Besides that, prototype of the proposed application is created to illustrate the concept of it.

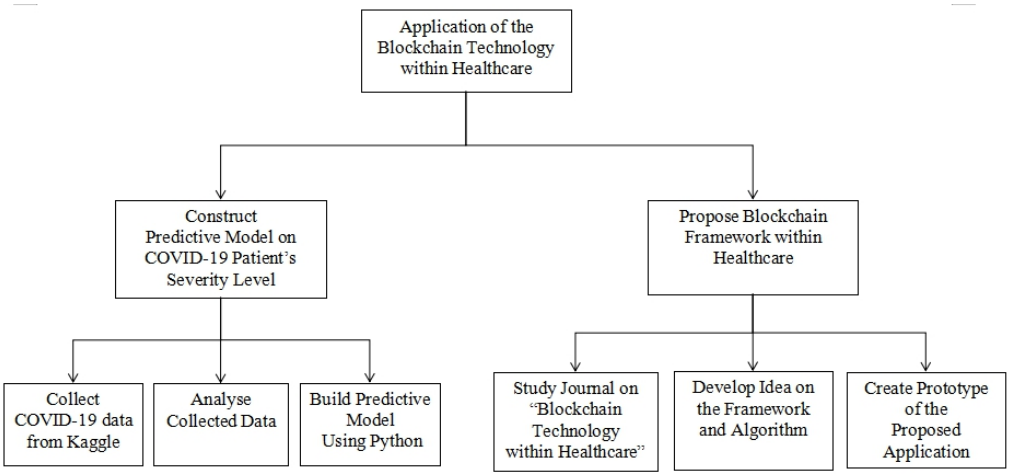


Figure 3.1: Project Planning

3.2 Collected COVID-19 Data

The COVID-19 dataset is collected from Kaggle, a popular platform that provides open source data for the area in data science and machine learning. The chosen dataset of COVID-19 titled "COVID-19 Symptoms Checker" is prepared by Bilal H. Hungund K., which he manually created the data referred

to the information and guidelines provided by World Healthcare Organization (WHO) and Ministry of Health and Family Welfare, India. The dataset included the symptoms of COVID-19 patients, countries that the patients have been visited, patients' severity level and other relevant information such as age and gender.

In details, the symptoms in this dataset consist of fever, tiredness, dry-cough, difficulty in breathing and sore-throat while the experiences consist of (muscle or joint) pains, nasal congestion, runny nose and diarrhea. According to WHO (2020), the common symptoms of COVID-19 are fever, dry cough and tiredness while difficulty in breathing and high temperature above 38°C are considered as the severe symptoms.

Besides that, the age data are categorized into 0-9, 10-19, 20-24, 25-59 and 60 and above while the gender data are categorized into female, male and transgender. With the given symptoms and relevant information, the severity level of infected patients are classified into none, mild, moderate and severe.

WHO advised people who faced severe symptoms should seek medical care and early treatments, especially elderly people with underlying diseases and weak immune systems. On the other hand, people who are young and face mild symptoms are advised to monitor their symptoms by staying at home, as most of them get to recover without hospitalization.

3.3 Proposed Models

In this project, Python is used to process the data and build a suitable model for it. Models such as K-Nearest Neighbors (KNN) and Logistic Regression (LR) are built based on the collected data and come out with the analysis on the severity level of COVID-19 patients.

KNN is a non-parametric method with its theory "similar inputs have similar outputs". Based on this theory, a test input x should be assigned the most common label among its k most similar training inputs (Liew, 2021). The KNN algorithm, given a positive integer k and an input x, finds the k points in the training data (x_i, y_i) that are closest to the x, represented by $N(x)$. KNN algorithm is able to perform its prediction for both classification and regression problems. Since the severity level of COVID-19 patient is

considered as a classification problem, the KNN algorithm makes the prediction as follows:

$$h(x) = \text{mode}(\{y'' : (x'', y'') \in N(x)\}) \tag{3.1}$$

$$P(Y = j \mid X = x) = \frac{1}{k} \sum_{x_i \in N(x)} I(y_i = j) \tag{3.2}$$

where

$h(x)$ = prediction of y corresponds to the input x

x'' = coordinate of training data x

y'' = coordinate of training data y

$N(x)$ = points which are close to input x

$P(Y = j \mid X = x)$ = probability of getting particular y prediction based on inputs x

x = input x

j = particular output y

k = number of closest neighbors/points to input x

The LR algorithm is a parametric method used for binary classification, with its assumption "the binary data are linearly separable with suitable parameters" (Liew, 2021). In contrast to the KNN algorithm, which predicts the output y directly, the LR algorithm measures the probability and makes predictions based on a cut-off value. The probability and prediction is made using the following functions :

$$\begin{aligned} &P(Y = 1 \mid X_1 = x_1, \cdots, X_p = x_p) \\ &= S(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p) \\ &= \frac{1}{1 + \exp(-(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p))} \end{aligned} \tag{3.3}$$

$$h(x) = \begin{cases} 1, P(Y = 1 \mid X = x) > p \\ 0, P(Y = 1 \mid X = x) \leq p \end{cases} \tag{3.4}$$

where

$P(Y=1 \mid X_1 = x_1, \cdots, X_p = x_p)$ = probability of getting prediction Y

x = inputs x

β_i = parameter constant corresponds to x_i

$h(x)$ = prediction of y corresponds to the probability obtained

p = optimal cut-off value

As the "COVID-19 Symptoms Checker" dataset has 4 severity levels, it is not a binary classification problem anymore. Hence, the general LR algorithm is derived to the following function in order to handle the 4-level qualitative target:

$$P(Y=1 \mid X=x) = \frac{1}{1+\sum_{i=2}^{K} e^{\beta_i x}}$$

$$P(Y=j \mid X=x) = \frac{e^{\beta_j x}}{1+\sum_{i=2}^{K} e^{\beta_i x}}, j = 2,3,4 \qquad (3.5)$$

where

$P(Y=1 \mid X=x)$ = probability of getting prediction $Y=1$ given input x

$P(Y=j \mid X=x)$ = probability of getting prediction $Y=j$ given input x

x = inputs x

β_i = parameter constant corresponds to x_i

k = the level of output y

β_j = parameter constant corresponds to x_i at the level j

j = the particular level k for predicted y

The following severity level of COVID-19 patient predictive model is integrated with the proposed application and available for all of the users to utilize it. As the symptoms of COVID-19 are differ from every infected person and COVID-19 test might not even detect the virus, doctors might miss out the subtle symptoms of the infected patient and misdiagnose. This could put the society into a high risk if anyone in contact with the infected patient. With the

constructed predictive model, doctors can improve on their diagnosis accuracy and give the patients with earlier treatments.

3.4 Proposed Framework

In this project, a consortium type of blockchain is being proposed and it is limited to those who have the authority to access the blockchain. The participants in this blockchain are divided into five groups which are patients, healthcare providers of the patients such as doctors and officers, other healthcare authorities such as medical researchers and health ministry, financial parties such as banking and insurance staffs, and Certificate Authority (CA).

In correspond to the second proposed problem statement stated in this project, the blockchain system is focused on the functions that can be carried out based on the patient records and EMR. The patient records and EMR are stored into the blockchain-based database in a secure manner and it can be viewed by every of the participants due to the blockchain's transparent property. In order for the participants to interact with the blockchain, a website application that connected to the blockchain is presented, enabling the participants to carry out their desired actions on the system.

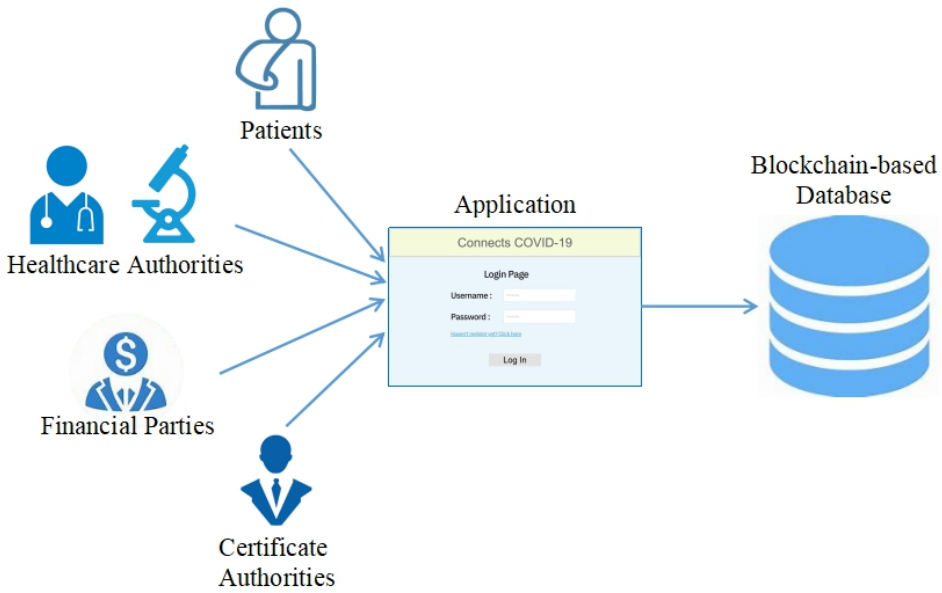


Figure 3.2: The Overview of Proposed Framework

3.4.1 Proposed Blockchain-based Database System

The data to be stored in the blockchain-based framework are the patient record and EMR that are mainly related to the COVID-19 matters. There is a database system that integrated with blockchain technology stores these data and it is linked to the proposed application. In general, there are 4 tables created in the database and stores the information of patient, doctor, hospital and EMR respectively.

The personal details of patients such as name, age, gender, contact number and guardian are stored under the table named “patient”. The disease and allergies of the patient are also stored for the reference of healthcare providers and authorities, as well as the symptoms of COVID-19 patients which is needed for building the predictive model.

The table named “emr” stores the records of patients whenever he or she visits the doctor at a hospital. The information stored include the visit date, doctor-in-charge, hospital name, title, description of the visit, prescription and any relevant image such as x-ray. The records are categorized into consultation and surgery, with their acronyms C and S to differentiate the EMR records stored in the table. These records are not limited to only COVID-19 but also the patients’ normal consultations and some of their main surgical records. With these information, healthcare providers are able to obtain the patient’s history medical records and come out with a suitable treatment based on the health condition of COVID-19 patient.

Besides that, the personal details of doctor-in-charge and contact number of hospital are also stored into the tables named “doctor” and “hospital”. Healthcare authorities such as researchers might want to know more information about the patients records and they can contact the particular doctor or hospital directly.

The information of patient, doctor and hospital are represented with an unique ID when a patient record is being stored. These IDs are the primary keys in their respective tables to uniquely identify them and act as the foreign keys in the “emr” table for reference purposes instead of duplicating the data. Nevertheless, each of the EMR is also assigned with a record ID automatically done by the database system.

A representation of the overall database design is shown with the diagram named Entity Relationship Diagram (ERD). The vertical rectangle is known as entity and represented as a table to store its data while the attributes within the entity reflect on the details of data required. Additionally, the diagram illustrates the relationship among the entities with their primary key and foreign keys are included. The ERD of the proposed blockchain-based database system is shown in Figure 3.3.

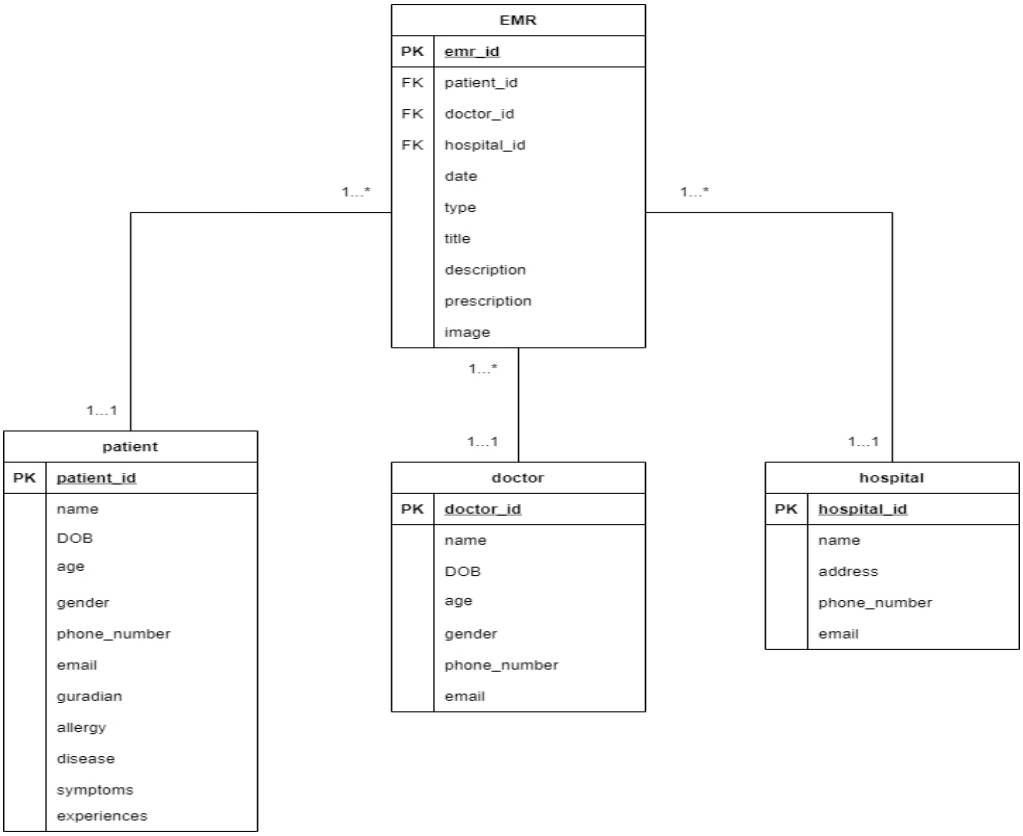


Figure 3.3: Entity Relationship Diagram (ERD)

3.4.2 Proposed Algorithms

Before carry out any of the actions on EMR, participants need to have the authorities on accessing the blockchain and CA deals with this matter. CA also acts as the admin with the job of removing or adding the blockchain participants. This is to ensure only the related people can view the data in blockchain and prevent hackers to access these sensitive data, and thus improve the data privacy. The overview of adding participants into blockchain process is shown in Figure 3.4.

The registration process is not only open for patients, as well as healthcare providers, healthcare authorities and financial parties who want to access the data in blockchain. With the registration process, they are added into the blockchain as participants and received ID, public key, private key and E-Certificate. In general, the public key is used to encrypt data while private key is used to decrypt data. For ID and E-Certificate, they represent the unique name and identity card of a blockchain participant respectively.

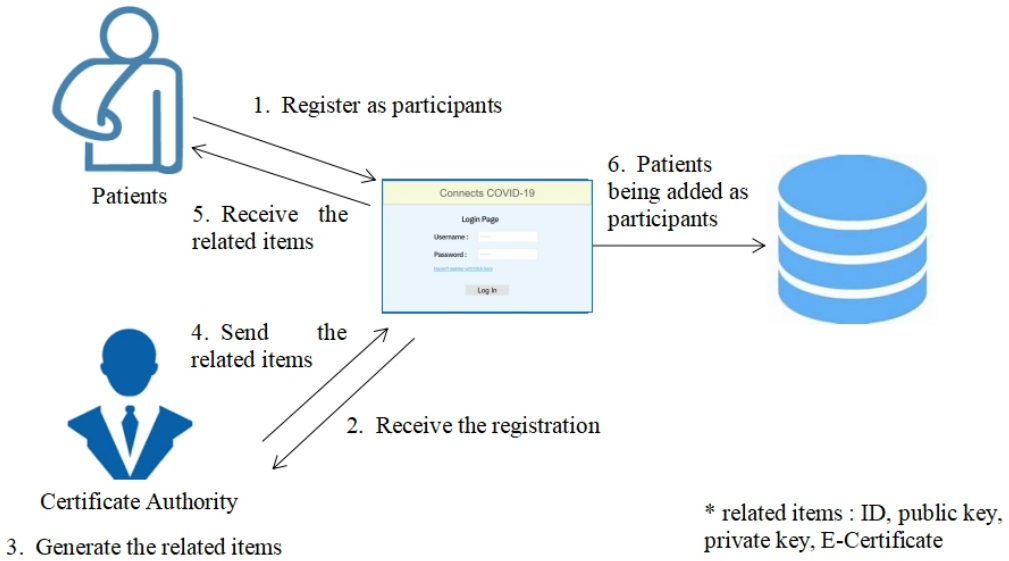


Figure 3.4: The Process of Adding Participants into Blockchain

After the patients meet with their doctors, their consultation details and diagnosis result need to be updated on the EMR and stored into the blockchain. Besides from consultation type of records, surgeries that are performed on the patients are also being recorded. Relevant images such as scanned X-ray or reference pictures are able to be uploaded into the blockchain. In this process, doctors need to have the permission from patients in order to update their patient medical records. The overview process of updating EMR into blockchain is shown in Figure 3.5.

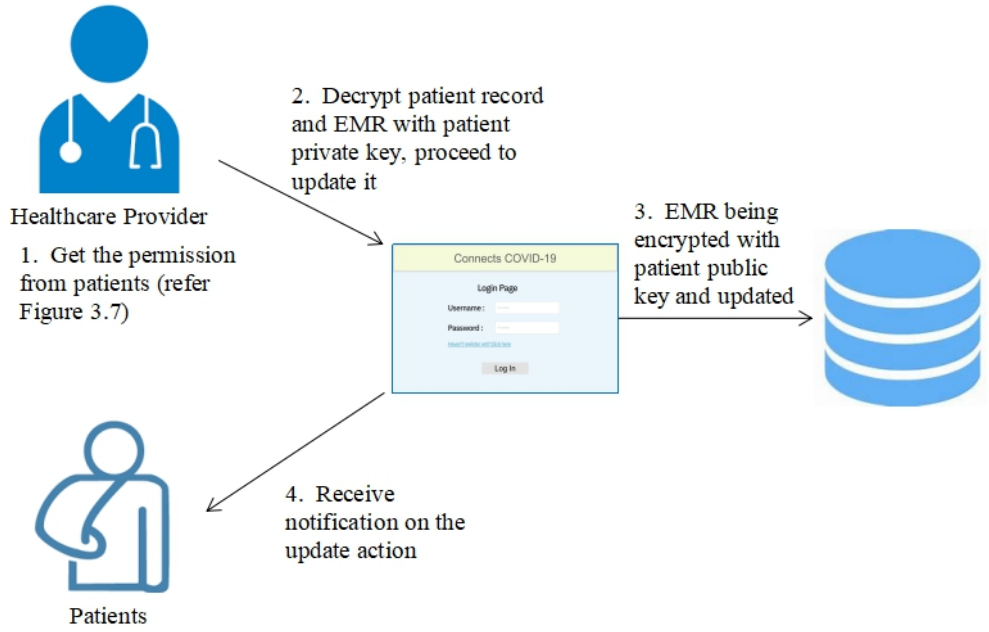


Figure 3.5: The Process of Updating Patient Records and EMR into Blockchain

As the healthcare authorities are part of the blockchain system, they are most probably would want to view the patient records and EMR in order to know their symptoms and treatment process. This could help them to have a better understanding on COVID-19 patients and help in their research on it. For the financial parties who are banking staff and insurance staff, they are the people who help patients in financial area by providing loans and selling COVID-19 package insurance. In order for them to verify the health condition of COVID-19 patients, they might want to view the relevant patient documents to proceed with their operation works.

Before viewing the patient records and EMR, healthcare authorities and financial parties need to gain the permission from patients in order to proceed with their action. The overview process of sharing patient records and EMR in blockchain is shown in Figure 3.6 by using healthcare authority as an example.

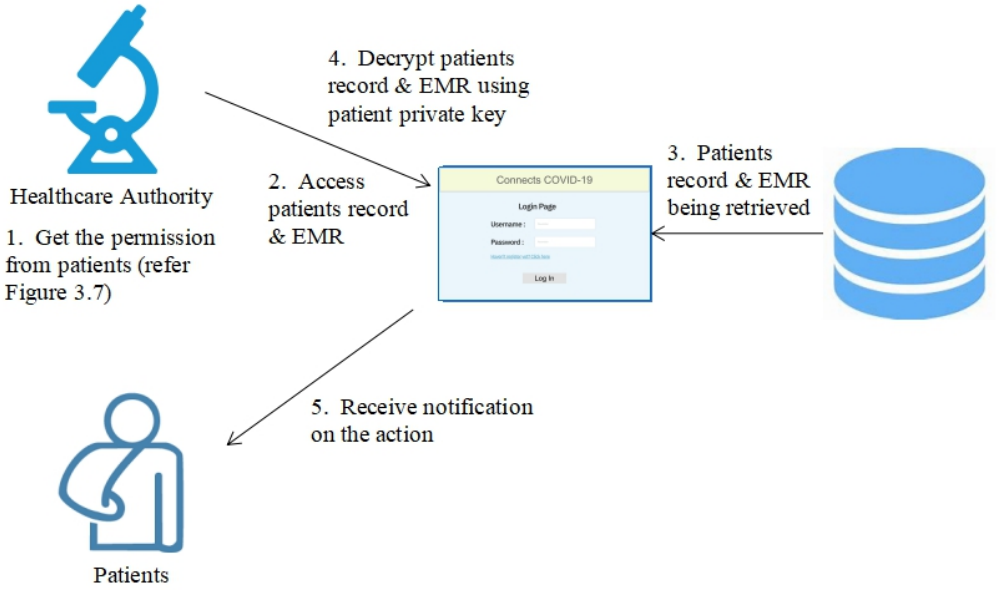


Figure 3.6: The Process of Sharing Patient Records and EMR in Blockchain

The patients in this blockchain system have full control of their own records accessibility. Anyone who wants to view the whole patient's record or update it, they need to ask permission from the patients. Since the patient's details and medical records are written under a particular hospital, viewers might also need to get the consent from healthcare providers as in hospital before asking the patients. The overview process of asking patients permission is shown in Figure 3.7.

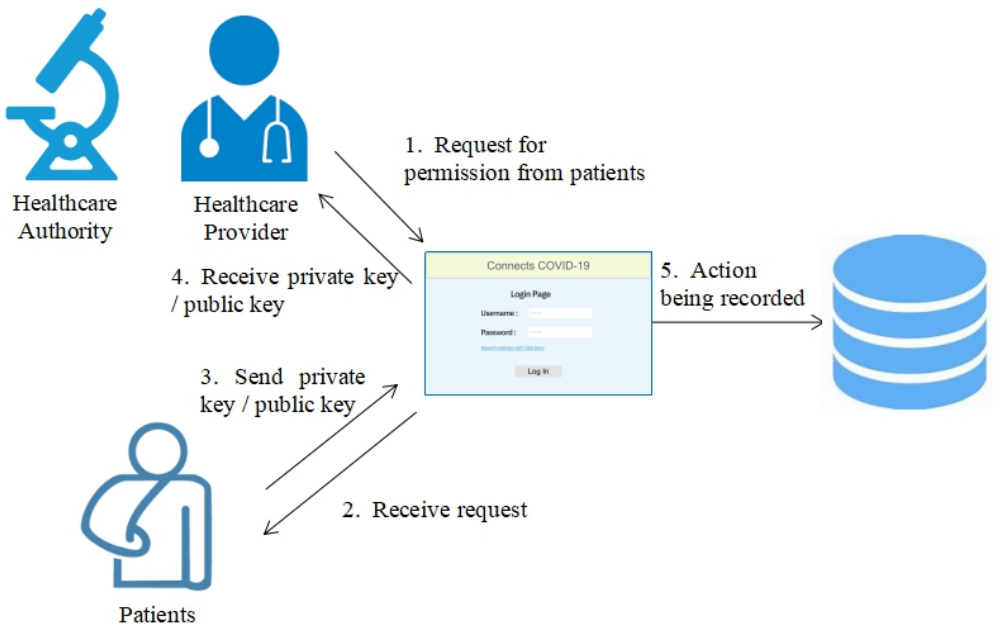


Figure 3.7: The Process of Asking Patient Permission

CHAPTER 4

SOFTWARE IMPLEMENTATION

4.1 Overview of the Software

As the constructed predictive model is related to the field of machine learning, Python is being chosen as the programming language to build the models in this project. Python is a high-level programming language that can be used for many purposes such as web development, software development and data science. It has always been the top choice for programmers in dealing with machine learning since it comes with large number of libraries including Tensorflow and Scikit-learn, which supports in the mentioned field.

In this project, the Python distribution known as Anaconda is installed, which is a data science tool that has gathered all the necessary and useful libraries needed for data science. The conda, package manager of Anaconda, helps to install all the packages and libraries at once and makes sure all the versions are compatible with each other in order to run a smooth program. The libraries that are used in this project includes:

(i) NumPy, a fundamental package that provides a collection of mathematical functions and supports in multi-dimensional array and matrix.

(ii) Pandas, a library that deals with data manipulation and data analysis based on data structure.

(iii) Matplotlib, a library that aids in plotting a variety of graphs from histogram to chart.

(iv) Seaborn, a data visualization library that built on top of matplotlib and generates more attractive graphs.

(v) Scikit-learn, a library that provides various functions for the whole process on performing machine learning and data science.

4.2 Software Approach

4.2.1 Exploratory Data Analysis (EDA)

To start off, the important libraries are being imported and data is loaded into the Jupyter notebook by stating its directory file path. A variable named “df” is declared to save the loaded data and perform upcoming commands on it. The first step after receiving the data is to carry out Exploratory Data Analysis (EDA). It is an initial approach to explore the basic characteristics of data and make some visualization based on the data summary. As the dataset is very large, it is difficult to extract the basic information from it by looking through every single collected data. Therefore, the dataframe in Python provides some of the functions to work on the data investigation and gives an insight of it.

The basic information of dataset to be extracted includes the number of observations, the features of the dataset, the values and data type of each feature and the existence of missing values. For function describe(), it summarizes the dataset by showing the minimum value, maximum value, mean, standard deviation and quartile of the data under each features. The software programme on performing these actions are attached under the Appendix C as reference.

Additionally, data visualization is carried out to give a clearer analysis on the data compared with just words description of it. It can be done by generating boxplot, distribution plot, histogram and chart. With the attractive graphs created by using Seaborn library, the data distribution such as skewness is clearly shown compared with just statistical data summary done by the function describe(). The interdependent relationship between the features are also able to be calculated by using the function corr() in Python.

4.2.2 Data Preprocessing

After figuring out all the basic information of the dataset, data preprocessing is needed before moving into the building model stage. It is a crucial step to organize the raw data in the way that it is suitable for building predictive models. If the data preprocessing is done correctly, it can improve the performance of predictive models and increase the result accuracy.

The missing values in dataset are common in real world situations and Python treats them as null values. Instead of deleting the whole observation

due to some of their values being absent, the usual way to overcome it is to replace them with mean, mode or median. The imputation named hot deck is also one of the solutions whereby the missing values is replaced with a similar observation from the dataset. Based on this "COVID-19 Symptoms Checker" dataset, there are no missing values hence value replacement is not needed.

Furthermore, some of the features are not related and do not contribute on building the predictive model, thus they are safe to be removed. This helps to reduce the time on running through all the data and increase the speed on building models. In this project, the objective is to build a predictive model to predict the COVID-19 patient's severity level based on their various symptoms. Therefore, the contact history and countries of the patient have been visited are not related to the objective and they are removed using the function drop().

A quick data analysis on the symptoms is done to make sure the patient's data with no symptoms (None_sympton=1) have the value of 0 on other symptoms features such as headache. The similar actions are done on the experiencing features. This is to ensure that the data are logical and acceptable, otherwise these weird data can be removed in order to increase the model accuracy.

Another common step in data preprocessing is feature transformation so that data is converted into a format which predictive models are able to understand it better. For categorical data, the one-hot encoding and ordinal encoding are used on nominal data and ordinal data respectively (Ng, 2020). These encoding methods transform the data into numerical format. The one-hot encoding helps to create dummy columns that store values 0 and 1 while the nominal data maps the values into integers such that 0, 1, 2 and so on (Ng, 2020). For numerical data, the common methods are min-max scaling and standard scaling to transform any skewed distribution close to normal distribution.

In this project's COVID-19 dataset, the one-hot encoding is performed on the Severity feature using function LabelEncoder(). Feature transformation on ages and genders are not needed as one-hot encoding results will be the same as the current dataset format. If the age feature has the continuous data instead of categorical data, it is better to perform feature scaling on it.

4.2.3 Model Development

Generally, there are two steps involved in constructing the model which are splitting the dataset into training set and testing set and building the desired predictive model by fitting in training set. The purpose of splitting dataset is to prepare the training set for optimizing the model and testing set for measuring the model performance.

In the Scikit-learn library, there is a function named train_test_split() that helps to perform the spitting automatically. Before that, the dataset is first separated into features data stored in variable 'x' and target data stored in variable 'y'. In this COVID-19 dataset, the column named Severity is separated out as the target data while the other columns are the features data. These variables are then put as parameters into train_test_split() function along with the random state fixed at 42. The parameter test_size indicates the proportion of splitting the data and the value 0.3 splits the training and testing set into 7:3.

An alternative way of performing the data splitting is stratified sampling, whereby the distribution of values in each column are uniform in both training and testing sets. The Scikit-learn library has the function named StratifiedShuffleSplit() that generates indexes and performs the similar process as the train_test_split().

After the training and testing set are ready, the predictive models are imported from the Scikit-learn library. The KNN predictive model is built by using KNeighborsClassifier() and it takes in the parameters n_neighbors and p, which represent the number of neighbors and power for Minkowski metric. There is another parameter named weights that assign the weight points with different calculations and then used for prediction. A total of 4 distinct KNN models are built with different combinations of p and weights parameters.

In constructing the LR models, LogisticRegression() is used to construct the default version of it while LogisticRegressionCV() is used to construct the L1 and L2 regularized version of it. The fitting of training set is done on both predictive models while the LogisticRegressionCV() takes in extra parameters to set the values for cross-validation folds, penalty that determines the regularization, multi-class that makes it as multinomial LR and

solvers that matches with the multi-class. The penalty value of 'l1' and 'l2' builds the Lasso regression and Ridge regression respectively.

4.2.4 Model Validation

The construction of predictive model did not ensure that it confirms give an accurate prediction, thus model validation is required to measure the model performances. The result obtained from model validation aids in decision making whether to accept or reject the predictive model. There is a function named accuracy_score() provides by Scikit-learn library which directly calculates the probability on getting the accurate predictions.

For KNN models, the 4 distinct models are put into a loop for building it with the n_neighbors parameter in the range from 1 to 10. The accuracy_score() is utilized to measure the model's prediction accuracy. A graph of the accuracy score changes with respect to n_neighbors parameter is plotted to easily find out the best value of n_neighbors by looking at the peak of graph. Among the 40 KNN models, the model with highest accuracy score is chosen as the best model along with its corresponding parameters.

The cross_val_score() is one of the common measurements to evaluate a model's performance. It is known as the cross-validation method that used to check the ability of a predictive model on predicting unseen data responding to different fitting dataset. The actual process of doing it is to split the dataset into n folds, each fold takes turn to be selected as training set while the others as testing sets. The chosen best KNN model is then put into the cross_val_score() as parameter along with cross-validation folds set at 4.

There are different model validation approaches for predictive models and the most common method is to generate the confusion matrix. Since the dataset of this project has 4 classes on its target named Severity, a multiple class error metrics is obtained by summarizing the actual results versus predicted results into a table. From the generated metrics, it derives the calculations of precision, recall, fscore, accuracy and Area Under Curve (AUC). These values are able to be computed by calling the functions score(), accuracy_score(), roc_auc_score() and confusion_matrix().

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Outcome of Constructed Model

5.1.1 Exploratory Data Analysis (EDA)

In this "COVID-19 Symptoms Checkers" dataset, there are a total of 316800 rows and 27 columns. The features that describe the data includes 5 different types of symptoms such as fever and tiredness, 4 different types of experiences such as pains and diarrhea, 5 categories of age, 3 categories of genders, 4 levels of severity, 3 types of contact status and countries that a patient have been visited.

All of the columns having integer as their data type except Country column has the object data type. This whole dataset is made up of complete data since there are no missing values indicated by the counts of null values is 0. Each of the columns contains binary values which are 0 and 1 representing the absence or existence of it while the Country column contains 10 different values of the country names. The occurrence of each value in their respective columns is shown in Table B-1.

The histogram of dataset is plotted, as shown in Appendix A, to reveal the distribution of numerical data. Since it contains only binary values, there are only 2 bins appeared at the 0 and 1 of x-axis and data distribution here is not helpful for visualizing the dataset. Therefore, a few countplots are generated on some of the features to visualize the number of observations. In general, only a minority of the patients have the symptoms of fever and sore-throat and experience pains and diarrhea. The common symptoms and experiences among the patients are dry cough, nasal congestion and runny nose while only 1/6 of patients having none of the stated symptoms and experiences. The remaining counterplots are not generated since they are even distributions as shown from the figures in Table B-1.

After the data preprocessing is carried out, the dataset left with only 20 columns as the features of contacts and country are removed and the severity level columns are combined into one column named Severity. The dataset is then ready for fitting into the predictive models. Before that, the correlation

between the features are computed and a graph of summarizing the frequency of correlation values is plotted in Graph A-11. From the histogram of absolute correlation values, it can be clearly seen that all of the correlation coefficients are under 0.5 and majority of them have the value 0.2.

5.1.2 Analysis of Constructed Model

Based on the accuracy score results obtained from the loop, it gives exactly the same value for all of the 4 distinct KNN models. The variation of parameters in building the KNN models using the particular dataset does not increase the performance of it. Since there is no difference among the KNN models, the default KNN model is chosen which holds the values '2' and 'uniform' for parameters p and weights respectively. The best value for n_neighbors parameter is 2 by identifying the peak of graph from Figure 5.1 as it yields the highest accuracy score.

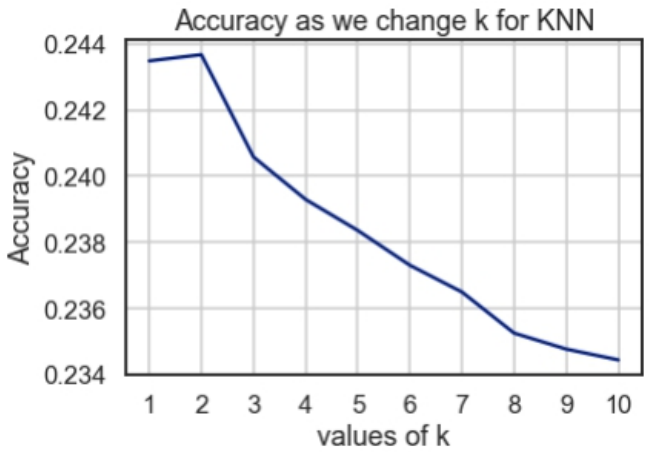


Figure 5.1: Accuracy Vs K Value

The measurement on the performance of chosen KNN model is done with the cross validation process. The results obtained from the folds of 4 are quite consistent such that they are vary by only 0.001 between 0.24 and 0.25. This indicates the performance of this KNN model is stable by producing the similar accuracy on predicting different unseen data.

Furthermore, the models' performance is evaluated by using the confusion matrix, which is also known as the contingency matrix in statistical field. It is a table that indicates the different predicted classes in rows and

different actual classes in columns while the cells represent the total occurrence of the particular condition. In this project, a total of 6 confusion matrices are generated to evaluate the chosen KNN model and various LR models.

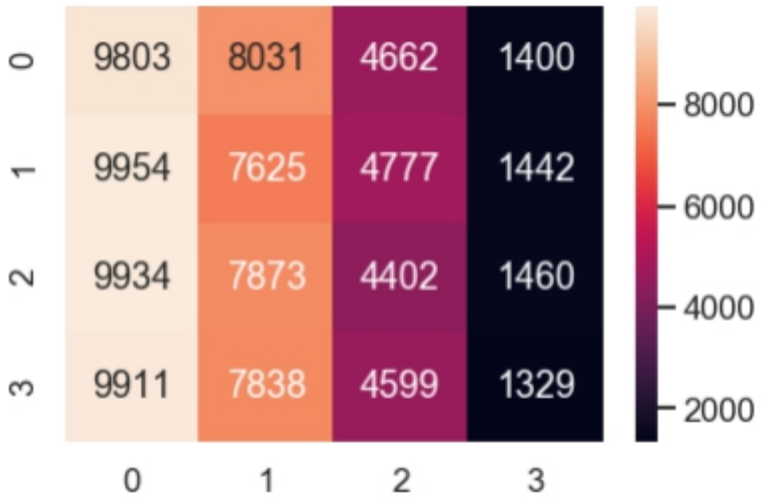


Figure 5.2: Confusion Matrix of KNN Model

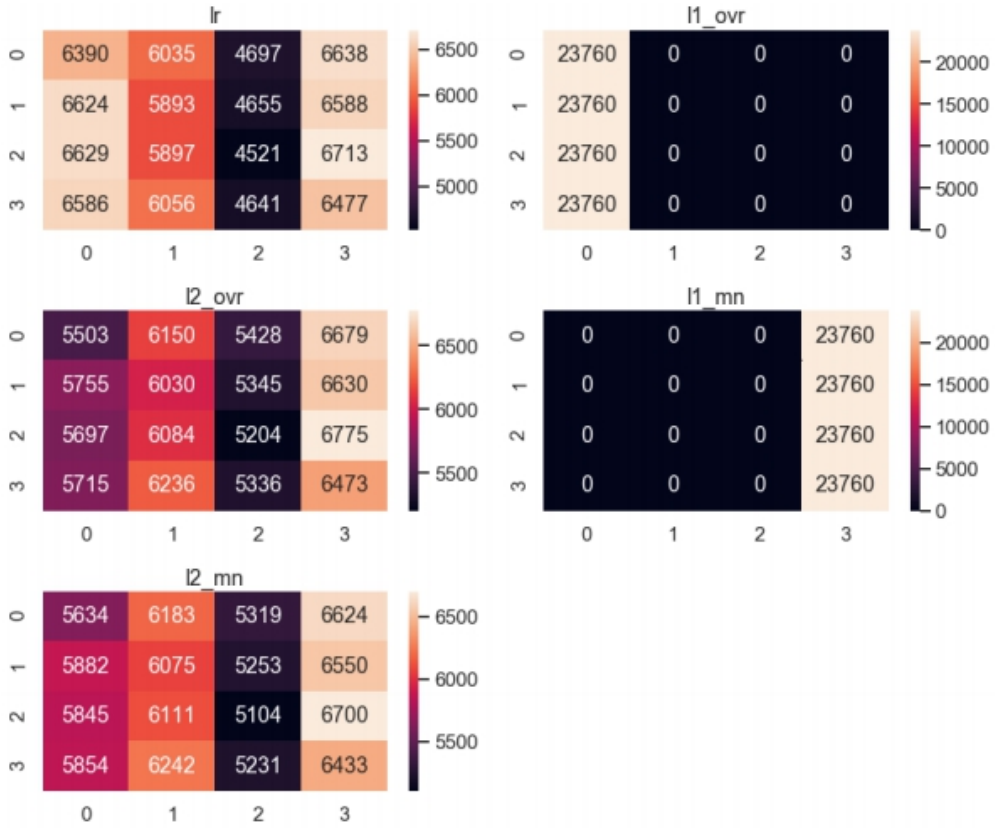


Figure 5.3: Confusion Matrices of 5 LR Models

The values obtained from confusion matrix are not sufficient enough to make inference hence some of them are extracted and used in other derived calculations. The computation of precision, recall, fscore, accuracy and AUC reflect the model's performance and they are the keys on selecting the best model. The stated AUC is only used on LR models since it gives the *p* optimal cut-off value for LR model prediction while KNN models do not use it in its algorithm. The interpretation on each of the key are defined as follows:

(i) The precision gives the proportion on capturing the correctly identified positive classes among the predicted positive results.

(ii) The recall, also known as the sensitivity, gives the proportion on capturing the correctly identified positive classes among all the actual positive classes.

(iii) Fscore is the harmonic mean of precision and recall which penalize the extreme values generated by inaccurate predictions.

(iv) Accuracy calculates the total correct predictions on both positive and negative classes among all the classes.

(v) AUC, stands for Area Under the Curves, measures the total area under the Receive Operating Characteristics (ROC) curve that plots the true positive rate versus false positive rate.

```
precision= 0.24135746919702217
recall= 0.2436763468013468
fscore= 0.22151733373763083
```

Figure 5.4: Performance Evaluation on Chosen KNN Model

	lr	l1_ovr	l2_ovr	l1_mn	l2_mn
precision	0.244944	0.0625	0.244194	0.0625	0.244548
recall	0.244960	0.2500	0.244213	0.2500	0.244592
fscore	0.243777	0.1000	0.243786	0.1000	0.244142
accuracy	0.244960	0.2500	0.244213	0.2500	0.244592
auc	0.496640	0.5000	0.496142	0.5000	0.496395

Figure 5.5: Performance Evaluation on 5 LR Models

Among 5 of the constructed LR models, it can be shown that the default LR and ridge regressions (l2_ovr and l2_mn) perform better based on

their higher precision and fscore values. For the other key values, they are roughly the same which differ by only 0.1 thus none of the inference can be made from here. By looking into the key values, the default LR model yields the best performance among the regression models as it has the highest overall accuracy on the various measurements.

Since there is only one predictive model being implemented into the application, accuracy score is taken into account on choosing the best constructed model. The default LR model is chosen since it has the higher accuracy of 0.24496, compared with the KNN model which has the average accuracy 0.24311. In other words, the default LR model is able to classify the COVID-19 patients into the respective 4 severity levels correctly with the higher accuracy 24.50% .

The higher precision, recalls and fscore of default LR model also reflects that it is a better model compared with the chosen KNN model. The results show that the default LR model is able to capture 24.49% of the positive predictions while 24.50% of them are predicted correctly. Prediction on the severity level of COVID-19 patients can be done by running the commands pred() that takes in both patient's symptoms and experiences as the inputs.

5.2 Proposed Application

For the related participants to interact with the blockchain-based database system and perform their desired actions, a website application is proposed to build up the connection between them. All the participants that are mentioned in the proposed framework are exactly the users of this proposed application. The proposed application is named as "Connects COVID-19" and there is a total of 8 main pages to meet with the users requirements.

When user first enters the website application, the login page is appeared and user is required to fill in their username and password. Users who are new to this blockchain system, they need to fill up their personal information and enter the referral code at the registration page. The referral code is provided only by the healthcare providers which acts as the "ticket pass" allowing users to register as part of the blockchain system. The registration is then approved by CA after checking on the validity of referral

code and add the newly registered user into the blockchain system as participant.

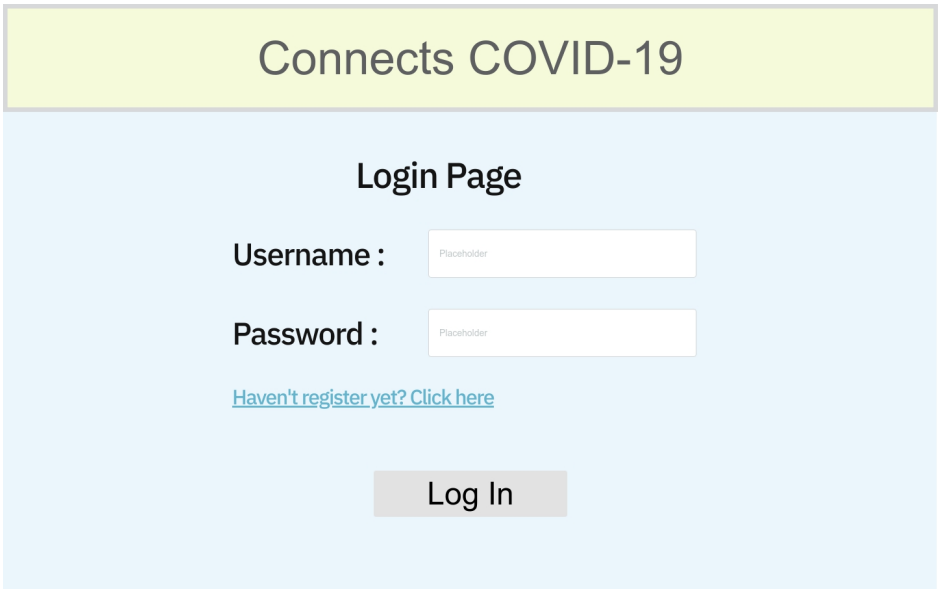


Figure 5.6: Login Page

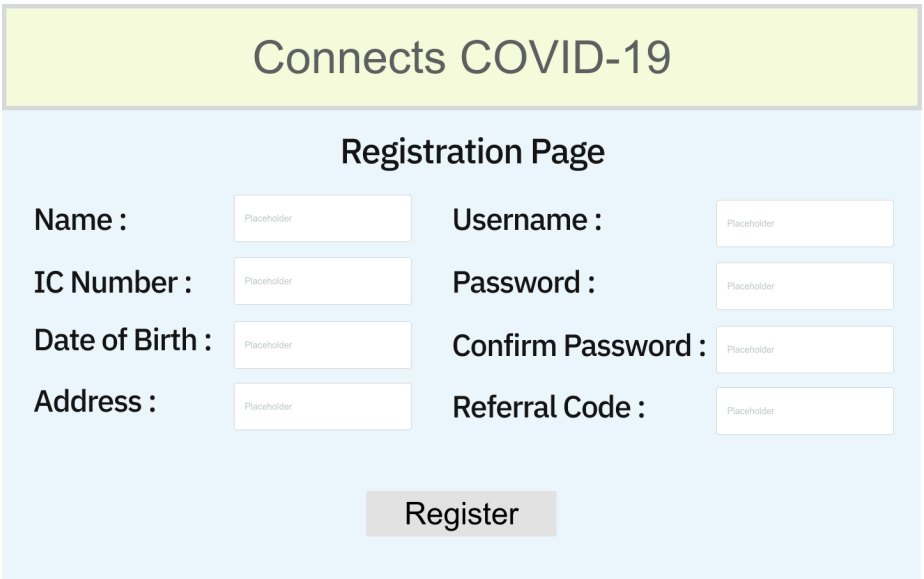


Figure 5.7: Registration Page

After successfully login, the home page of application displays the user's status, profile and some action buttons which are applicable to the user. The status of users is differentiated into 7 types, which are patient, doctor, hospital staff, researcher, government staff, banking staff and insurance staff. For the action buttons, they are designed to meet with the user actions

including view EMR, update EMR and request for permission on viewing EMR. In addition, there is a bell button attached at the top right corner and a list of notifications are displayed if the user clicks on it. System sends the notifications to user if someone accesses his EMR or requests for permission on accessing it.

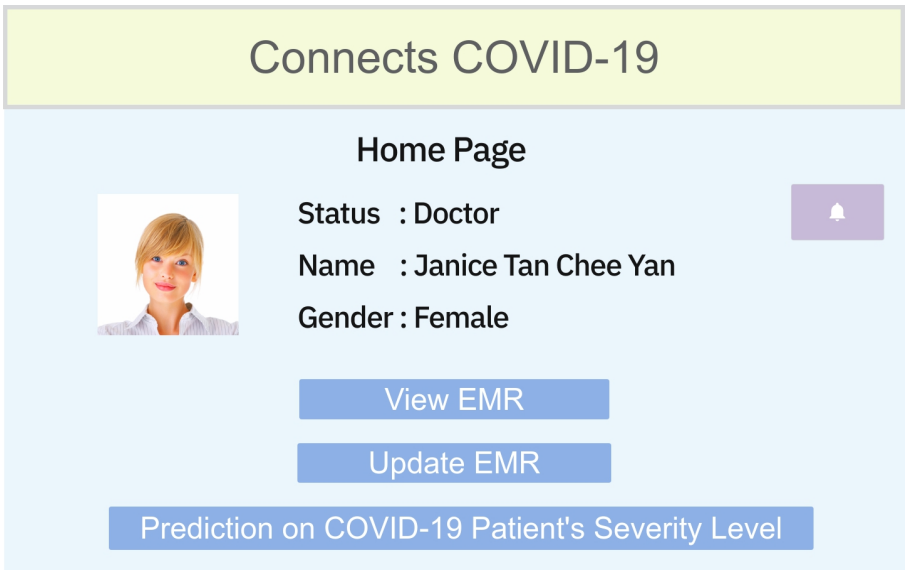


Figure 5.8: Home Page

On the webpage of "view EMR", the application displays with 2 drop-down lists and asks for user to select for the particular hospital and patient ID. After clicking on the search button, all of the EMR lists are retrieved from the system by displaying only the patient ID, EMR ID, date and title in ascending rows with respect to the patient's ID. By clicking on the EMR ID, the patient's EMR are fully displayed if the user has obtained the consent from patient on accessing it while patient is automatically given the permission to view on its own EMR.

A button with "request for permission" is appeared on right side of the record row if user does not have the permission to view it but wishes to do so. There is also a download button designed at the left top corner and users have the option to download all the documents as PDF or CSV file.

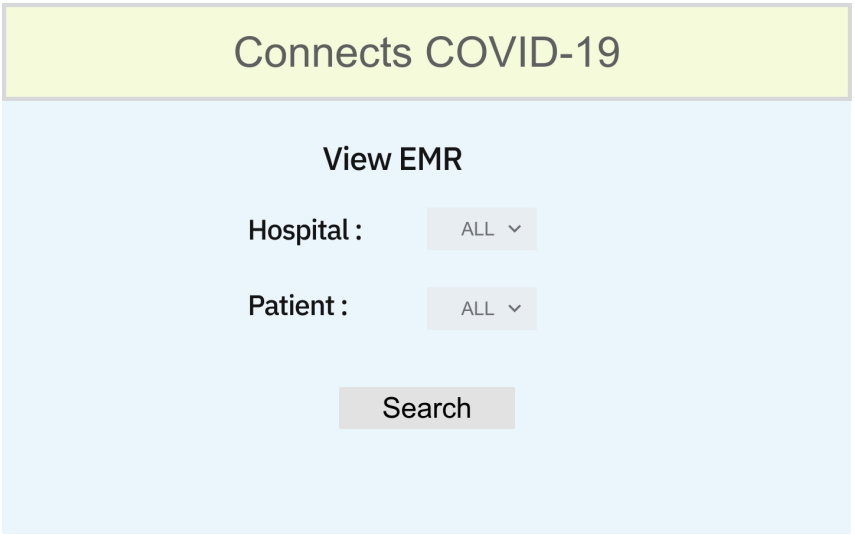


Figure 5.9: View EMR Page

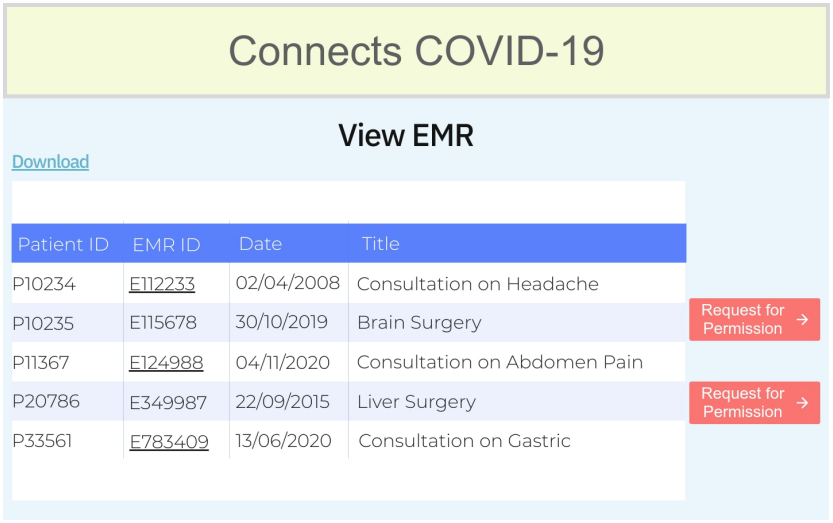


Figure 5.10: View EMR Page After Selecting Hospital and Patient

For user whose status is doctor, he is able to access the “update EMR” webpage to create and store a new patient’s medical record into the system. The application requests the user to fill in the hospital ID, doctor ID, patient name, date, title, description of the consultation or surgery and prescription. An EMR is successfully updated into the blockchain-based database system after the submit button is being clicked.

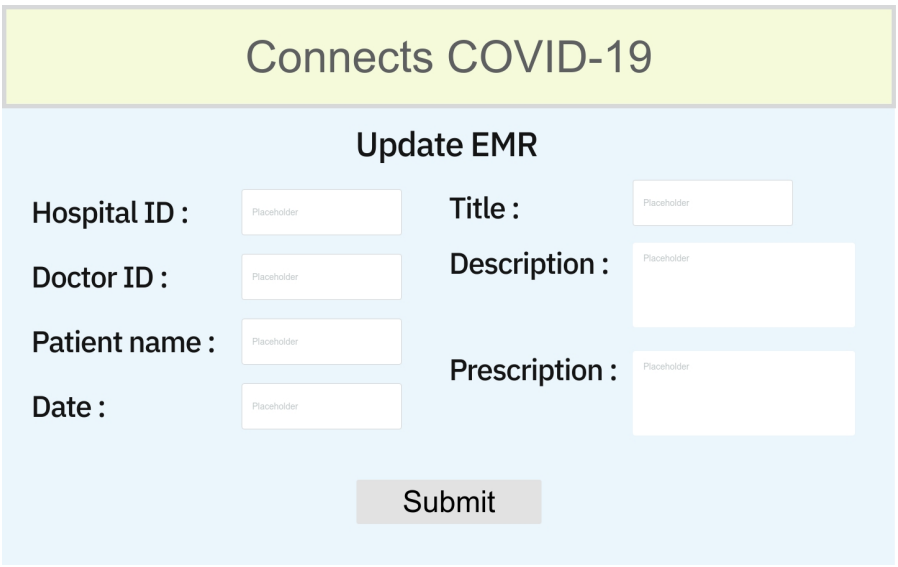


Figure 5.11: Update EMR Page

The application also included a webpage named “prediction on COVID-19 patient’s severity level” which integrated with the constructed LR predictive model and displays the outcome from it to the users. The webpage is similar to the “view EMR” but it displays the patients’ personal details and it has an additional checkbox for users to select which particular patients to be chosen. Then, the predictive model takes in the patients’ data and predicts his severity level based on his symptoms.

With the predicted severity level of COVID-19 patients, doctors are able to make more precise diagnosis by cross checking its own diagnosis. Then, suitable treatment can be given to COVID-19 patients and helped them to recover in a shorter period. Besides that, researchers can take the predictive model outcome into consideration and assists in their research process. Other than that, financial parties including bank and insurance staff are able to make more precise decisions on the approval of loan and COVID-19 insurance plans based on the predicted severity level.

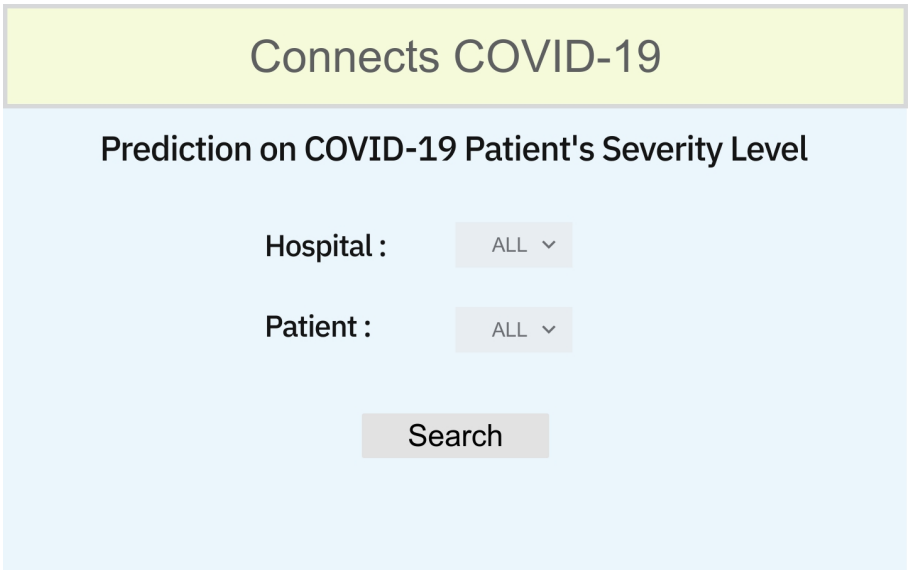


Figure 5.12: Prediction on COVID-19 Patient's Severity Level Page

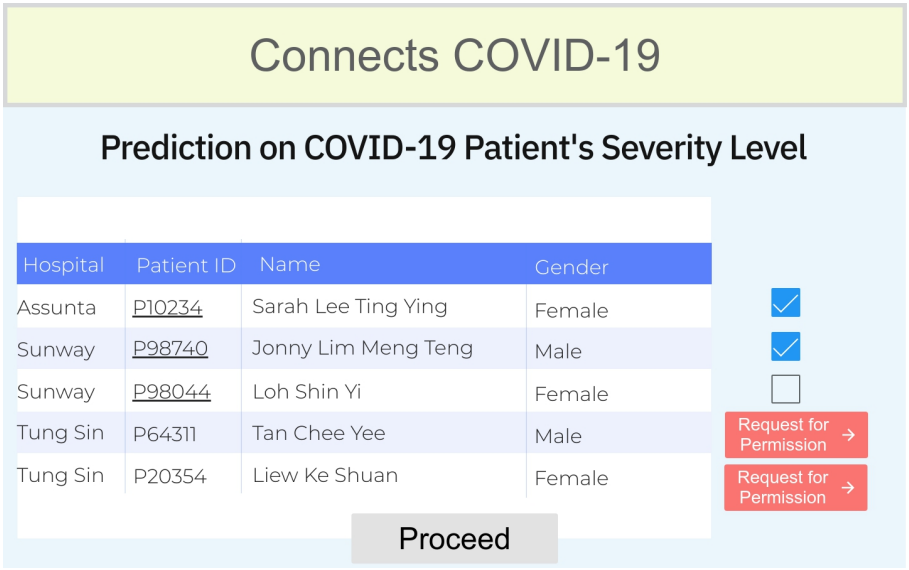


Figure 5.13: Prediction on COVID-19 Patient's Severity Level Page After Selecting Hospital and Patient

5.3 Challenges on Model Development

Despite the default LR model being chosen as the best model to be implemented, its accuracy score from different assessments are all below 0.5, making it considered as a poor performance model theoretically. An ideal predictive model should yield an accuracy score of at least 0.8, with the best being near 0.99. The low accuracy of the constructed predictive model is most likely due to the data that was chosen.

"COVID-19 Symptoms Checker" dataset was manually generated by the author using information from the official websites of health organizations and associations. It did not contain any true data collected from actual COVID-19 patient and thus the resulted accuracy is not ideal. The actual patients' medical records and data are difficult to obtain since they are most probably kept in the hospital as confidential documents. The predictive model would be able to produce a more precise result such that if the real world patient data is taken.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

In this study, a predictive model named Logistic Regression is built to predict the COVID-19 patients' severity level based on their various symptoms. The processes of EDA, data preprocessing, model development and model validation are carried out in order to construct a reliable predictive model that gives accurate results. The constructed Logistic Regression model is then integrated with the proposed application aids in the users' decision making based on the predicted results.

Furthermore, a multi-function framework is proposed with the implementation of blockchain technology. Firstly, people who work in the healthcare industry or involved in the COVID-19 field, are figured out and identified as the blockchain system participants. Next, a total of 4 algorithms are proposed, in corresponds to the stated functions in the second objective. The proposed blockchain-based framework is able to store, retrieve and update the EMR and utilize the COVID-19 patient data to give prediction on his severity level. Other than that, an application is proposed to connect the participants to the blockchain-based database, not to mention the prototype that presents the concept of it.

Additionally, the proposed blockchain-based application that stores EMR brings convenience in sharing data, such that users can directly access the patient data without the involvement of third parties. Activities that involve data sharing includes researchers who want to use the data for COVID-19 vaccine development or reference on medical records when patients transfer from a hospital to another hospital. This could save a lot of time and increase the overall productivity in healthcare industry. Patients are also given the controls on their data in the way that other people can only access it using the private key with patient's consent.

6.2 Recommendations for Future Work

In order to improve the performance of predictive model, the actual patient data should be taken and fitted into the model. The dataset should include more details of the patients such as the subtle COVID-19 symptoms and any underlying diseases faced by them. The subtle symptoms including reduced appetite and rashes and the underlying disease including diabetes and kidney diseases might indicate or cause the COVID-19 patients at a more severe condition. By taking the actual patient data, the predictive model is able to "learn" from the data pattern and produce a more accurate prediction on the COVID-19 patient's severity level.

In addition, the actual patient data might have too many features and this could affect the model performances such that increases the time on building it and produces a lower accuracy score. It is a good practice to check on the correlations of the features before proceed to the model development. In this project, all of the absolute correlations have the value lower than 0.5 thus whole dataset is used without any further actions. However, for the actual patient data case, it is better to remove the features which have high absolute correlation values. This indicates that the features are somewhat linearly dependent with the other features and they do not give much contributions for prediction. Therefore, features which have the absolute correlation values of above 0.8 are safe to remove for the sake of model performances.

Moreover, the proposed application could be further enhanced by working on its interoperability. The application in this project is proposed as a website and can only be used through browsers. The future research could be done on developing the proposed framework into mobile applications that can run on various operating systems, including Android, iOS and HarmonyOS, integrated with the blockchain technology. With the mobile applications, all types of mobile phone users are able to access the EMR at anywhere and anytime.

REFERENCES

Agbo, C.C., Mahmoud, Q.H. and Eklund, J.M., 2019. *Blockchain technology in healthcare: a systematic review.* [online] Available at: <https://www.mdpi.com/2227-9032/7/2/56/htm> [Accessed 15 July 2020]

Angraal, S., Krumholz, H.M. and Schulz, W.L., 2017. *Blockchain technology: applications in health care.* [online] Available at: <https://www.ahajournals.org/doi/full/10.1161/circoutcomes.117.003800> [Accessed 15 July 2020]

Christ, M.J., Nikolaus Permana Tri, R., Chandra, W. and Guanwan, W., 2019. *Exploring Blockchain in Healthcare Industry. 2019 International Conference on ICT for Smart Society (ICISS),* [e-journal] pp. 1-4. Available through: Universiti Tunku Abdul Rahman Library website <https://library.utar.edu.my/> [Accessed 12 August 2020]

Drescher,D., 2017. *Blockchain basics : a non-technical introduction in 25 steps.* Berkeley, California: Apress.

Flynt, O., 2016. *Blockchain : the ultimate guide to understanding the hidden economy*. San Bernardino, CA.

Frankenfield, J., 2019. *Bitcoin.* [online] Available at: <https://www.investopedia.com/terms/b/bitcoin.asp> [Accessed 9 June 2020]

Hölbl, M., Kompara, M., Kamišalić, A. and Nemec Zlatolas, L., 2018. *A systematic review of the use of blockchain in healthcare.* [online] Available at : <https://www.mdpi.com/2073-8994/10/10/470/htm> [Accessed 16 July 2020]

Kuo, T.T., Kim, H.E. and Ohno-Machado, L., 2017. *Blockchain distributed ledger technologies for biomedical and health care applications.* [online] Available at : <https://academic.oup.com/jamia/article/24/6/1211/4108087> [Accessed 29 July 2020]

Lee, A.R., Kim, M.G. and Kim, I.K., 2019. *SHAREChain: Healthcare data sharing framework using Blockchain-registry and FHIR. 2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM),* [e-journal] pp. 1087-1090. Available through: Universiti Tunku Abdul Rahman Library website <https://library.utar.edu.my/> [Accessed 13 August 2020]

Li, X., Jiang, P., Chen, T., Luo, X. and Wen, Q., 2017. *A survey on the security of blockchain systems. Future Generation Computer Systems.* [online] Available at: <http://dx.doi.org/10.1016/j.future.2017.08.020> [Accessed 30 July 2020]

Liew, H.H., 2021. *Predictive Model kNN.* [lecture note] Sungai Long: Univiersiti Tunku Abdul Rahman.

Liew, H.H., 2021. *Predictive Model Logistic Regression.* [lecture note] Sung ai Long: Univiersiti Tunku Abdul Rahman.

Ng, O.E., 2020. *Optimizing Data for Supervised Learning.* [lecture note] Sungai Long: Universiti Tunku Abdul Rahman.

Ni, W., Huang, X., Zhang, J. and Yu, R., 2019. *HealChain: A Decentralized Data Management System for Mobile Healthcare Using Consortium Blockchain. 2019 Chinese Control Conference (CCC),* [e-journal] pp. 6333-6338. Available through: Universiti Tunku Abdul Rahman Library website <https://library.utar.edu.my/> [Accessed 12 August 2020]

Susie, N. and Aylin, W., 2020. *A comprehensive timeline of the coronacvirus pandemic at 1 year, from China's first case to the present.* [online] Available at : <https://www.businessinsider.com/coronavirus-pandemic-timeline-history-major-events-2020-3> [Accessed 18 March 2021]

Tanwar, S., Parekh, K. and Evans, R., 2020. *Blockchain-based electronic healthcare record system for healthcare 4.0 applications. Journal of Information Security and Applications*, [e-journal] 50, p.102407. Available through: Universiti Tunku Abdul Rahman Library website <https://library.utar.edu.my/> [Accessed 11 August 2020]

Usman, M. and Qamar, U., 2020. *Secure Electronic Medical Records Storage and Sharing Using Blockchain Technology. Procedia Computer Science*, [e-journal] 174, pp.321–327. Available through: Universiti Tunku Abdul Rahman Library website <https://library.utar.edu.my/> [Accessed 11 August 2020]

Vora, J., Nayyar, A., Tanwar, S., Tyagi, S., Kumar, N., Obaidat, M.S. and Rodrigues, J.J.P.C., 2018. *BHEEM: A Blockchain-Based Framework for Securing Electronic Health Records. 2018 IEEE Globecom Workshops (GC Wkshps)*, [e-journal] pp. 1-6. Available through: Universiti Tunku Abdul Rahman Library website <https://library.utar.edu.my/> [Accessed 10 August 2020]

Wang, S., Wang, J., Wang, X., Qiu, T., Yuan, Y., Ouyang, L., Guo, Y. and Wang, F.-Y., 2018. *Blockchain-Powered Parallel Healthcare Systems Based on the ACP Approach*. *IEEE Transactions on Computational Social Systems,* [e-journal] 5(4), pp.942–950. Available through: Universiti Tunku Abdul Rahman Library website <https://library.utar.edu.my/> [Accessed 10 August 2020]

Watanabe, H., Fujimura, S., Nakadaira, A., Miyazaki, Y., Akutsu, A. and Kishigami, J.J., 2015. *Blockchain contract: A complete consensus using blockchain.* [e-journal] Available at : <https://ieeexplore.ieee.org/document/7398721> [Accessed 30 July 2020]

World Heath Organization (WHO), n.d.. *Coronavirus.* [online] Available at: <https://www.who.int/health-topics/coronavirus#tab=tab_1> [Accessed 4 April 2021]

World Heath Organization (WHO), 2020. *Coronavirus disease (COVID-19).* [online] Available at: <https://www.who.int/news-room/q-a-detail/coronavirus-disease-covid-19> [Accessed 4 April 2021]

World Heath Organization (WHO), 2020. *Coronavirus disease (COVID-19) in Malaysia.* [online] Available at : <https://www.who.int/malaysia/emergencies/coronavirus-disease-(covid-19)-in-malaysia> [Accessed 2 June 2020]

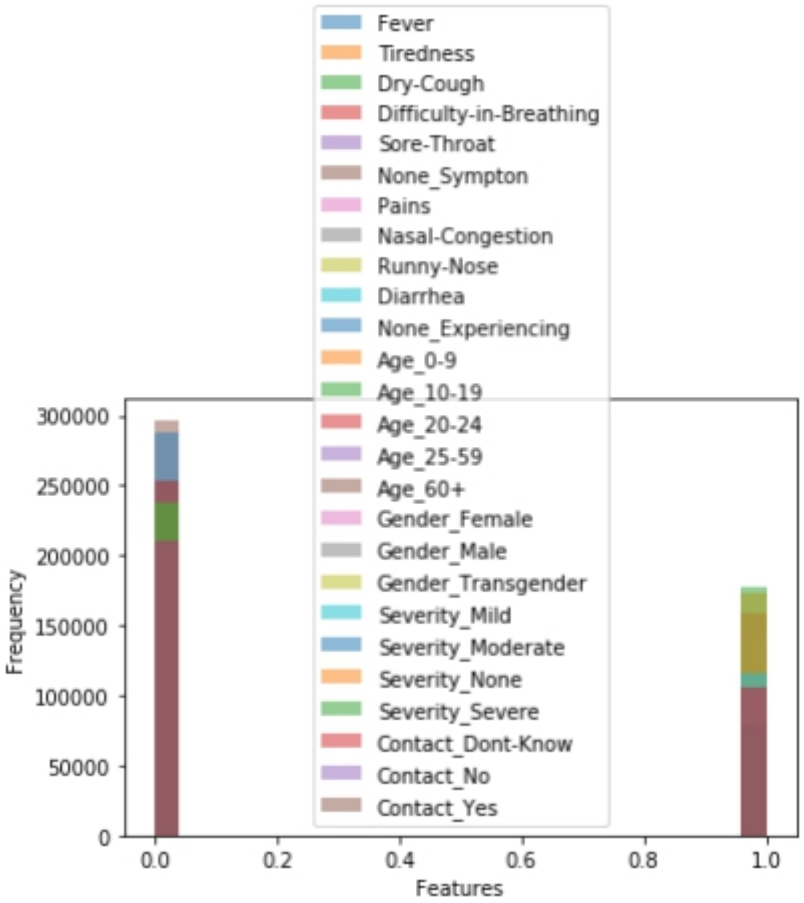
World Heath Organization (WHO), 2020. *WHO COVID-19 Dashboard.* [online] Available at: <https://covid19.who.int/> [Accessed 17 March 2021]

Zubaydi, H.D., Chong, Y.W., Ko, K., Hanshi, S.M. and Karuppayah, S., 2019. *A Review on the Role of Blockchain Technology in the Healthcare Domain.* [online] Available at: <https://www.mdpi.com/2079-9292/8/6/679/htm> [Accessed 17 July 2020]

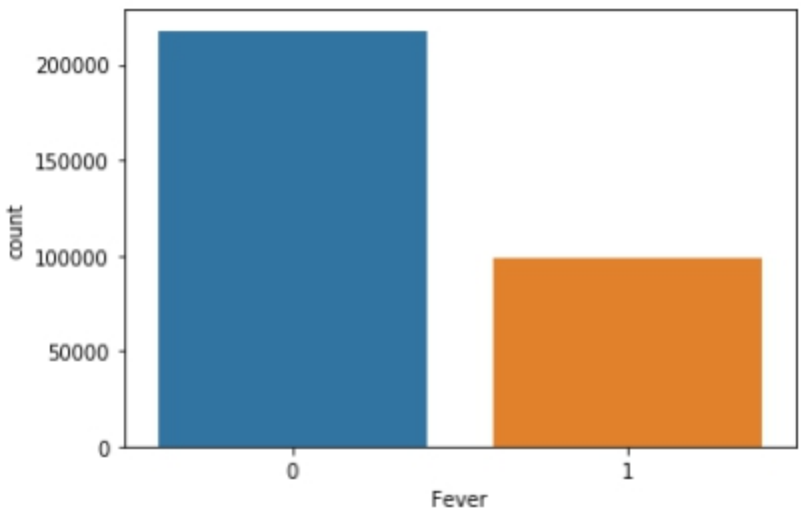
Zyskind, G., Nathan, O. and Pentland, A. “Sandy” , 2015. *Decentralizing Privacy: Using Blockchain to Protect Personal Data.* [e-journal] Available at : <https://ieeexplore.ieee.org/document/7163223> [Accessed 30 July 2020]

APPENDICES

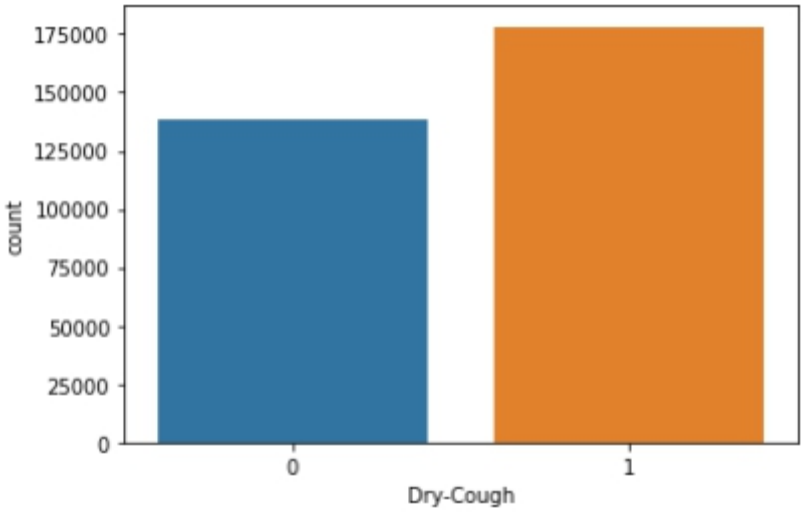
APPENDIX A: Graphs



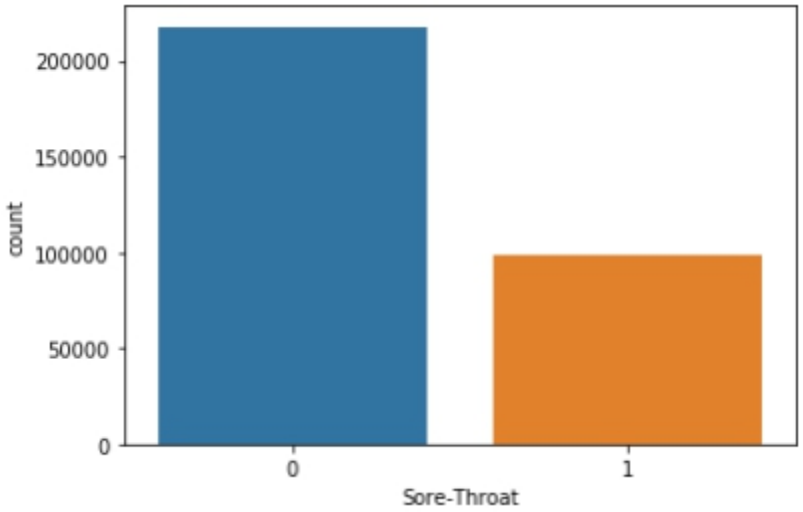
Graph A-1 : Histogram of Dataset



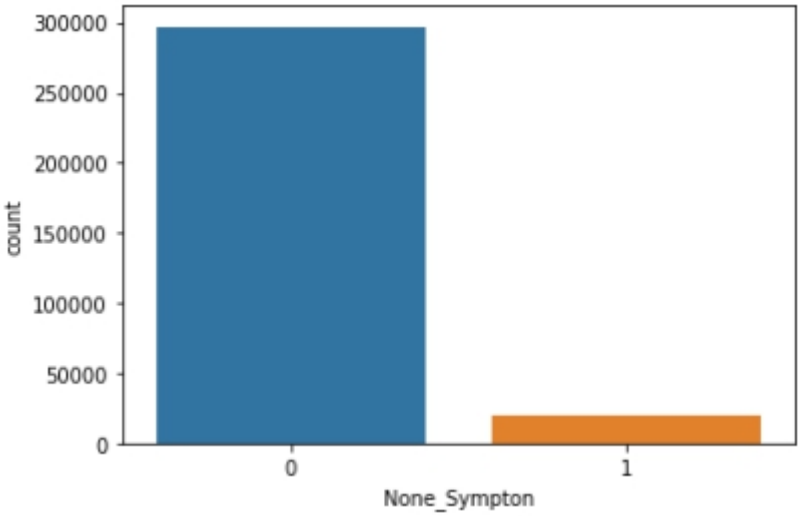
Graph A-2 : Counterplot of Fever Column



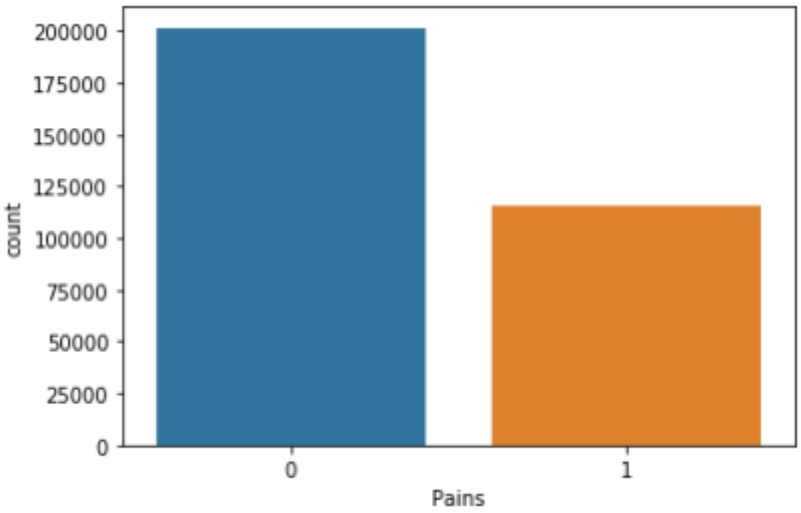
Graph A-3 : Counterplot of Dry-Cough Column



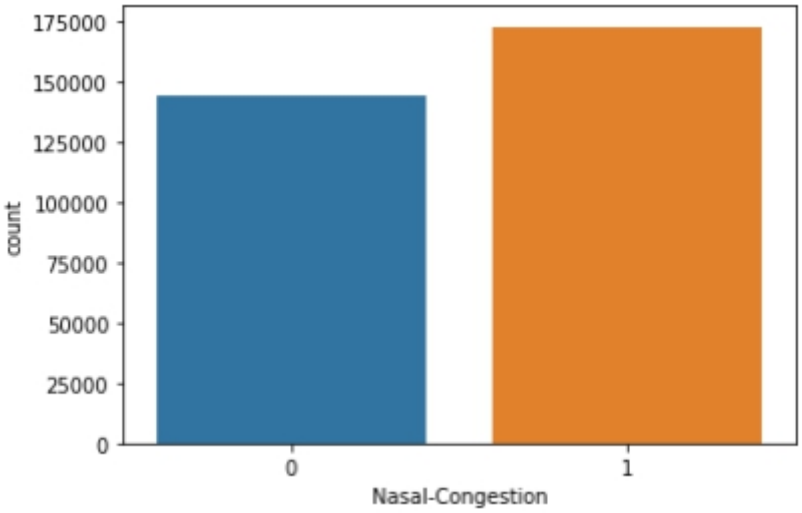
Graph A-4 : Counterplot of Sore-Throat Column



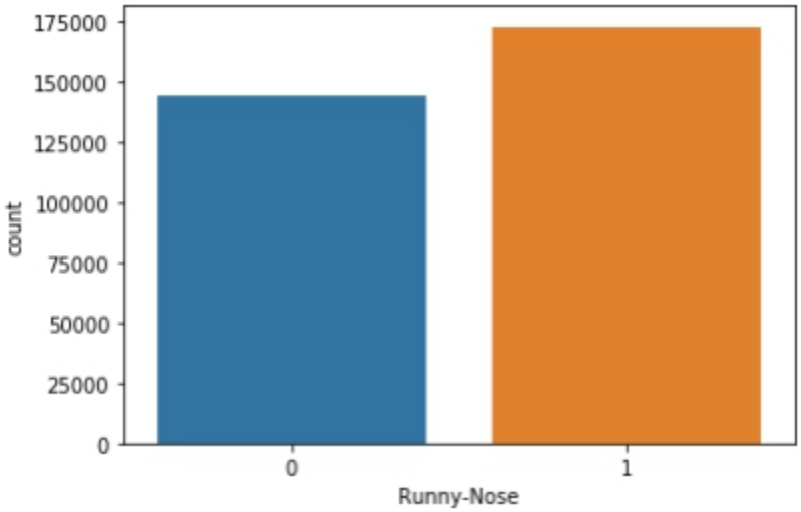
Graph A-5 : Counterplot of None_Sympton Column



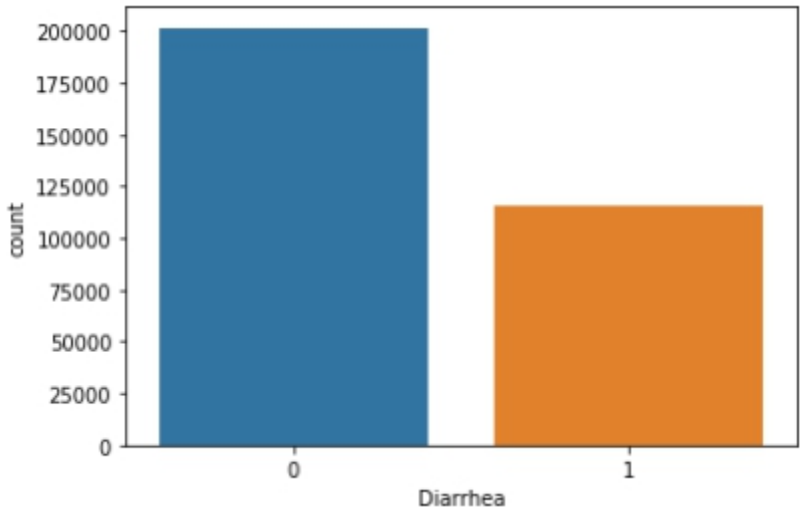
Graph A-6 : Counterplot of Pains Column



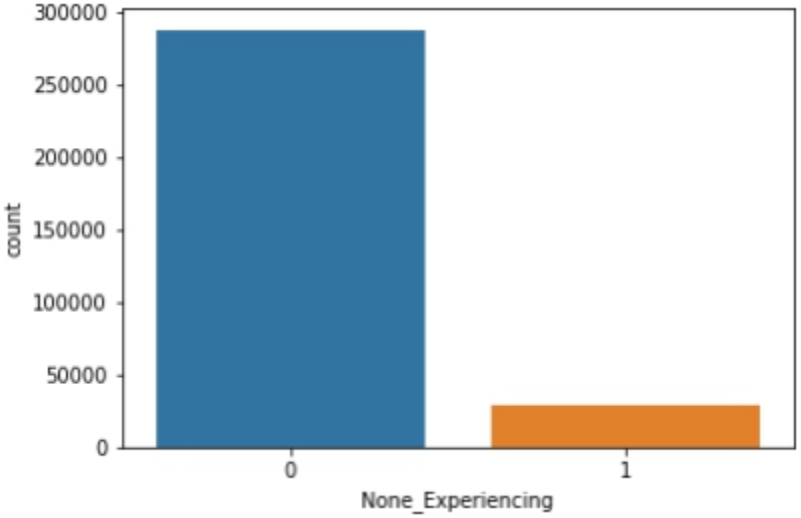
Graph A-7 : Counterplot of Nasal-Congestion Column



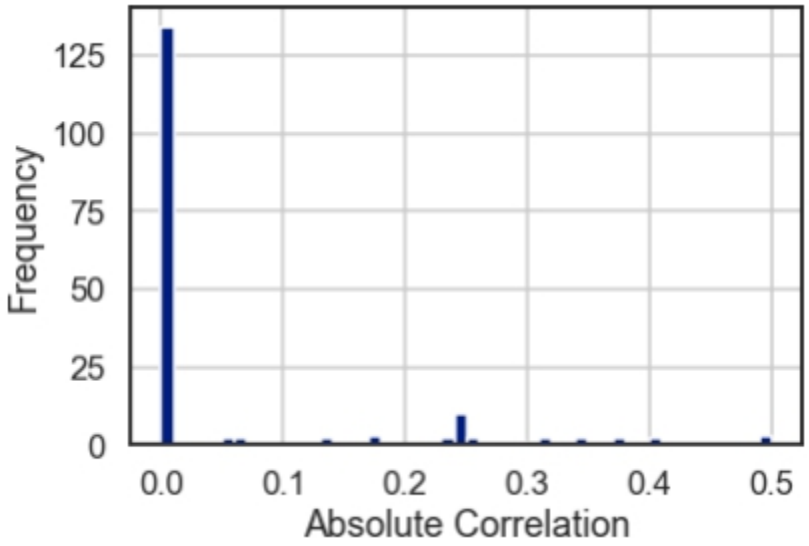
Graph A-8 : Counterplot of Runny-Nose Column



Graph A-9 : Counterplot of Diarrhea Column



Graph A-10 : Counterplot of None_Experiencing Column



Graph A-11 : Frequency of Absolute Correlation Values

APPENDIX B: Tables

Table B-1 : Occurance of Values in Each Column

Column	Data type	Values with its occurancce
Fever	Integer	0 : 217800 1 : 99000
Tiredness	Integer	0 : 158400 1 : 158400
Dry-Cough	Integer	0 : 138600 1 : 178200
Difficulty-in-Breathing	Integer	0 : 158400 1 : 158400
Sore-Throat	Integer	0 : 217800 1 : 99000
None_Sympton	Integer	0 : 297000 1 : 19800
Pains	Integer	0 : 201600 1 : 115200
Nasal-Congestion	Integer	0 : 144000 1 : 172800
Runny-Nose	Integer	0 : 144000 1 : 172800
Diarrhea	Integer	0 : 201600 1 : 115200
None_Experiencing	Integer	0 : 288000 1 : 28800
Age_0-9	Integer	0 : 253440 1 : 63360
Age_10-19	Integer	0 : 253440 1 : 63360
Age_20-24	Integer	0 : 253440 1 : 63360

Age_25-29	Integer	0 : 253440 1 : 63360
Age_60+	Integer	0 : 253440 1 : 63360
Gender_Female	Integer	0 : 211200 1 : 105600
Gender_Male	Integer	0 : 211200 1 : 105600
Gender_Transgender	Integer	0 : 211200 1 : 105600
Severity_Mild	Integer	0 : 237600 1 : 79200
Severity_Moderate	Integer	0 : 237600 1 : 79200
Severity_None	Integer	0 : 237600 1 : 79200
Severity_Severe	Integer	0 : 237600 1 : 79200
Contact_Dont-Know	Integer	0 : 211200 1 : 105600
Contact_No	Integer	0 : 211200 1 : 105600
Contact_Yes	Integer	0 : 211200 1 : 105600
Country	Object	Italy : 31680 China : 31680 UAE : 31680 Other-EUR : 31680 Spain : 31680 Iran : 31680 Other : 31680 Republic of Korean : 31680 Germany : 31680

		France : 31680

APPENDIX C: Python Codes and Outputs

Load data

In [1]:

```
# import all the needed libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:

```
# load the data into Python notebook
filepath = "covid_symptoms/Cleaned-Data.csv"
df = pd.read_csv(filepath)
```

Data basic information / Exploratory Data Analysis (EDA)

In [3]:

```
# first of all, have an insight of the data
df.head()
```

Out[3]:

	Fever	Tiredness	Dry-Cough	Difficulty-in-Breathing	Sore-Throat	None_Sympton	Pains	Nasal-Congestion	Runny-Nose	Di
0	1	1	1	1	1	0	1	1	1	
1	1	1	1	1	1	0	1	1	1	
2	1	1	1	1	1	0	1	1	1	
3	1	1	1	1	1	0	1	1	1	
4	1	1	1	1	1	0	1	1	1	

5 rows × 27 columns

In [4]:

```
# check rows and columns
df.shape
```

Out[4]:

```
(316800, 27)
```

In [5]:

```
# check columns
df.columns.tolist()
```

Out[5]:

```
['Fever',
 'Tiredness',
 'Dry-Cough',
 'Difficulty-in-Breathing',
 'Sore-Throat',
 'None_Sympton',
 'Pains',
 'Nasal-Congestion',
 'Runny-Nose',
 'Diarrhea',
 'None_Experiencing',
 'Age_0-9',
 'Age_10-19',
 'Age_20-24',
 'Age_25-59',
 'Age_60+',
 'Gender_Female',
 'Gender_Male',
 'Gender_Transgender',
 'Severity_Mild',
 'Severity_Moderate',
 'Severity_None',
 'Severity_Severe',
 'Contact_Dont-Know',
 'Contact_No',
 'Contact_Yes',
 'Country']
```

In [6]:

```
# check non-null values and datatypes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 316800 entries, 0 to 316799
Data columns (total 27 columns):
 #   Column                    Non-Null Count   Dtype 
---  ------                    --------------   ----- 
 0   Fever                     316800 non-null  int64 
 1   Tiredness                 316800 non-null  int64 
 2   Dry-Cough                 316800 non-null  int64 
 3   Difficulty-in-Breathing   316800 non-null  int64 
 4   Sore-Throat               316800 non-null  int64 
 5   None_Sympton              316800 non-null  int64 
 6   Pains                     316800 non-null  int64 
 7   Nasal-Congestion          316800 non-null  int64 
 8   Runny-Nose                316800 non-null  int64 
 9   Diarrhea                  316800 non-null  int64 
 10  None_Experiencing         316800 non-null  int64 
 11  Age_0-9                   316800 non-null  int64 
 12  Age_10-19                 316800 non-null  int64 
 13  Age_20-24                 316800 non-null  int64 
 14  Age_25-59                 316800 non-null  int64 
 15  Age_60+                   316800 non-null  int64 
 16  Gender_Female             316800 non-null  int64 
 17  Gender_Male               316800 non-null  int64 
 18  Gender_Transgender        316800 non-null  int64 
 19  Severity_Mild             316800 non-null  int64 
 20  Severity_Moderate         316800 non-null  int64 
 21  Severity_None             316800 non-null  int64 
 22  Severity_Severe           316800 non-null  int64 
 23  Contact_Dont-Know         316800 non-null  int64 
 24  Contact_No                316800 non-null  int64 
 25  Contact_Yes               316800 non-null  int64 
 26  Country                   316800 non-null  object
dtypes: int64(26), object(1)
memory usage: 65.3+ MB
```

In [7]:

```
# another way to check is there any null values
df.isnull().values.any()
```

Out[7]:

```
False
```

In [8]:

```
# to check which columns having the null values
df.isnull().sum()
```

Out[8]:

```
Fever                        0
Tiredness                    0
Dry-Cough                    0
Difficulty-in-Breathing      0
Sore-Throat                  0
None_Sympton                 0
Pains                        0
Nasal-Congestion             0
Runny-Nose                   0
Diarrhea                     0
None_Experiencing            0
Age_0-9                      0
Age_10-19                    0
Age_20-24                    0
Age_25-59                    0
Age_60+                      0
Gender_Female                0
Gender_Male                  0
Gender_Transgender           0
Severity_Mild                0
Severity_Moderate            0
Severity_None                0
Severity_Severe              0
Contact_Dont-Know            0
Contact_No                   0
Contact_Yes                  0
Country                      0
dtype: int64
```

In [9]:

```
# have an insight on the data values
df.describe()
```

Out[9]:

	Fever	Tiredness	Dry-Cough	Difficulty-in-Breathing	Sore-Throat	None_Symp
count	316800.000000	316800.000000	316800.000000	316800.000000	316800.000000	316800.000
mean	0.312500	0.500000	0.562500	0.500000	0.312500	0.062
std	0.463513	0.500001	0.496079	0.500001	0.463513	0.242
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
50%	0.000000	0.500000	1.000000	0.500000	0.000000	0.000
75%	1.000000	1.000000	1.000000	1.000000	1.000000	0.000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000

8 rows × 26 columns

In [10]:

```
# get the total number of unique values on each columns/fields
df.nunique()
```

Out[10]:

```
Fever                        2
Tiredness                    2
Dry-Cough                    2
Difficulty-in-Breathing      2
Sore-Throat                  2
None_Sympton                 2
Pains                        2
Nasal-Congestion             2
Runny-Nose                   2
Diarrhea                     2
None_Experiencing            2
Age_0-9                      2
Age_10-19                    2
Age_20-24                    2
Age_25-59                    2
Age_60+                      2
Gender_Female                2
Gender_Male                  2
Gender_Transgender           2
Severity_Mild                2
Severity_Moderate            2
Severity_None                2
Severity_Severe              2
Contact_Dont-Know            2
Contact_No                   2
Contact_Yes                  2
Country                     10
dtype: int64
```

In [11]:

```
# get the unique values for each columns/fields
col = df.columns.tolist()
for i in col:
    print(i, df[i].unique())
```

```
Fever [1 0]
Tiredness [1 0]
Dry-Cough [1 0]
Difficulty-in-Breathing [1 0]
Sore-Throat [1 0]
None_Sympton [0 1]
Pains [1 0]
Nasal-Congestion [1 0]
Runny-Nose [1 0]
Diarrhea [1 0]
None_Experiencing [0 1]
Age_0-9 [1 0]
Age_10-19 [0 1]
Age_20-24 [0 1]
Age_25-59 [0 1]
Age_60+ [0 1]
Gender_Female [0 1]
Gender_Male [1 0]
Gender_Transgender [0 1]
Severity_Mild [1 0]
Severity_Moderate [0 1]
Severity_None [0 1]
Severity_Severe [0 1]
Contact_Dont-Know [0 1]
Contact_No [0 1]
Contact_Yes [1 0]
Country ['China' 'Italy' 'Iran' 'Republic of Korean' 'France' 'Spain' 'Germa
ny'
 'UAE' 'Other-EUR' 'Other']
```

In [12]:

```
# get the number of each unique values in each columns/fields
col = df.columns.tolist()
for i in col:
    print(df[i].value_counts())
    print("\n")
```

```
0    217800
1     99000
Name: Fever, dtype: int64


1    158400
0    158400
Name: Tiredness, dtype: int64


1    178200
0    138600
Name: Dry-Cough, dtype: int64


1    158400
0    158400
Name: Difficulty-in-Breathing, dtype: int64
```

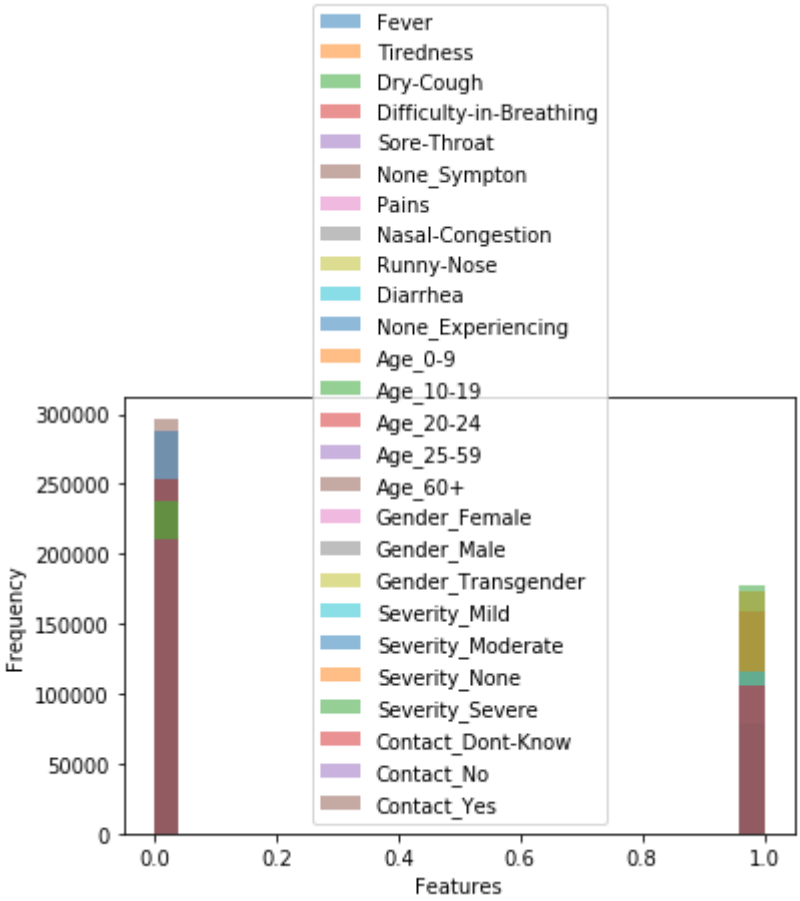
Plotting graphs for data visualization

In [13]:

```
# plot histogram for each of the numeric data type columns into 1 graph
ax = df.plot.hist(bins=25, alpha=0.5)
ax.set_xlabel('Features')
```

Out[13]:

```
Text(0.5, 0, 'Features')
```

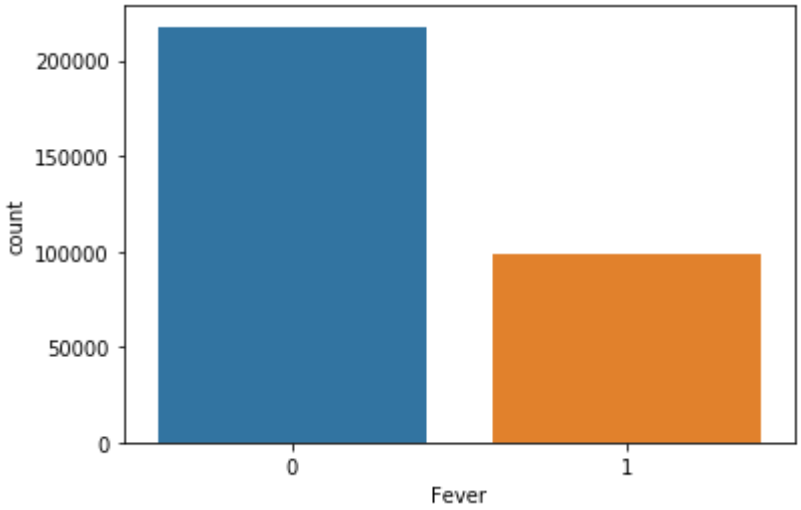


In [14]:

```
# plot the individual histogram for better visualizing
sns.countplot(data=df, x = 'Fever')
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1363e720f48>
```

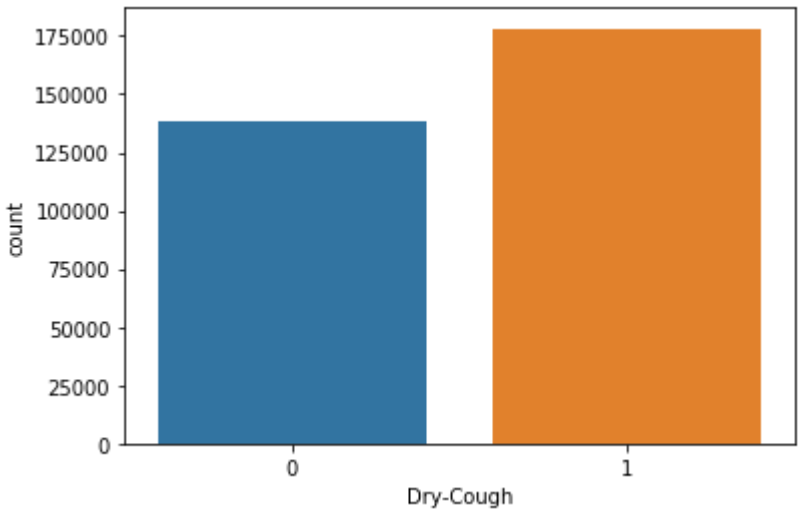


In [15]:

```
sns.countplot(data=df, x = 'Dry-Cough')
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1363d4f6888>
```

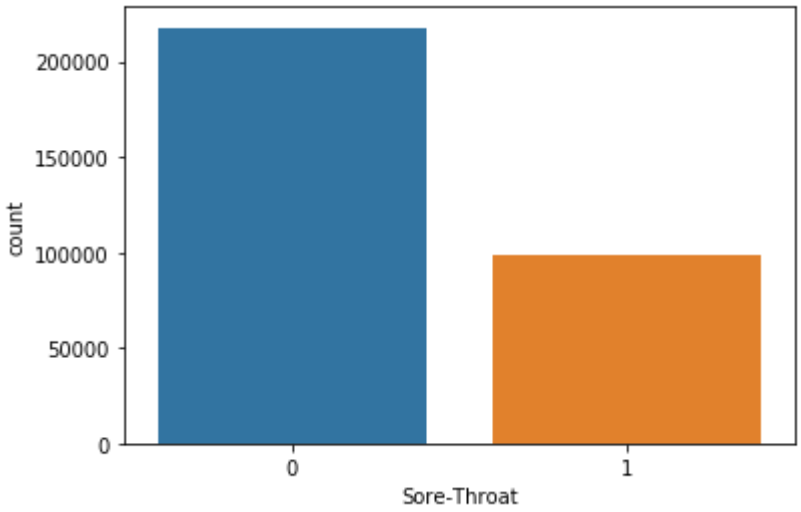


In [16]:

```
sns.countplot(data=df, x = 'Sore-Throat')
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1363d553cc8>
```

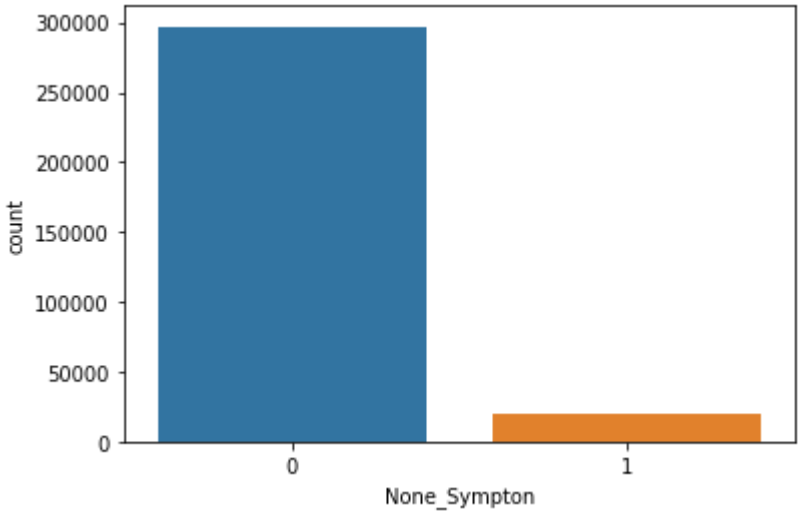


In [17]:

```
sns.countplot(data=df, x = 'None_Sympton')
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1363d5d0a08>
```

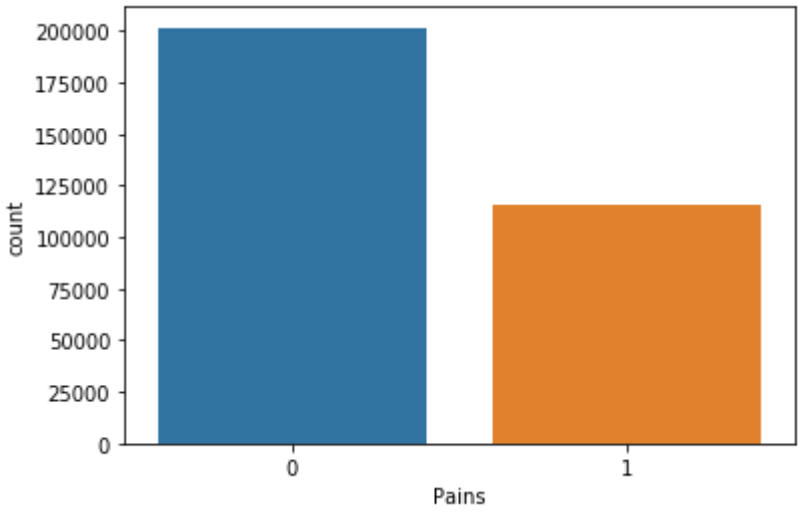


In [18]:

```
sns.countplot(data=df, x = 'Pains')
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1363d627148>
```

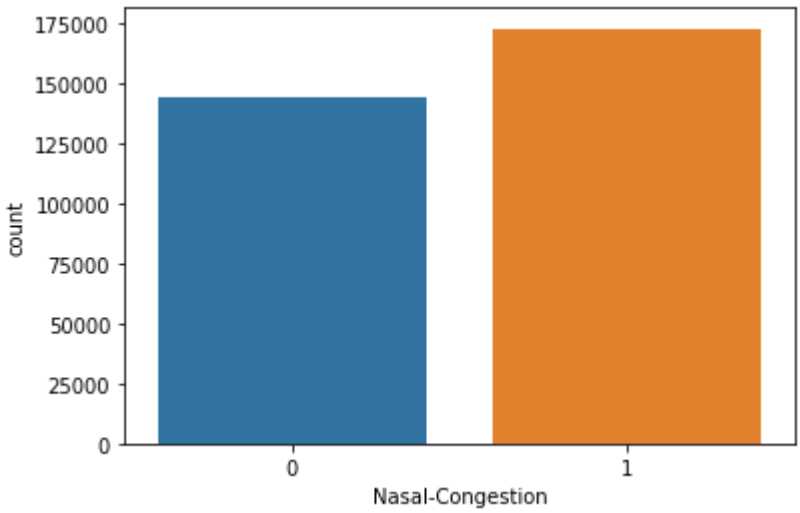


In [19]:

```
sns.countplot(data=df, x = 'Nasal-Congestion')
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1363d680248>
```

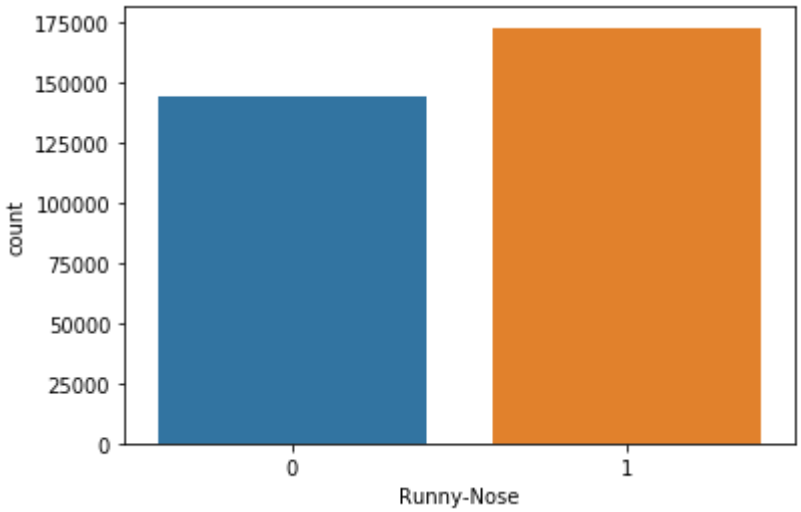


In [20]:

```
sns.countplot(data=df, x = 'Runny-Nose')
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1363e107d48>
```

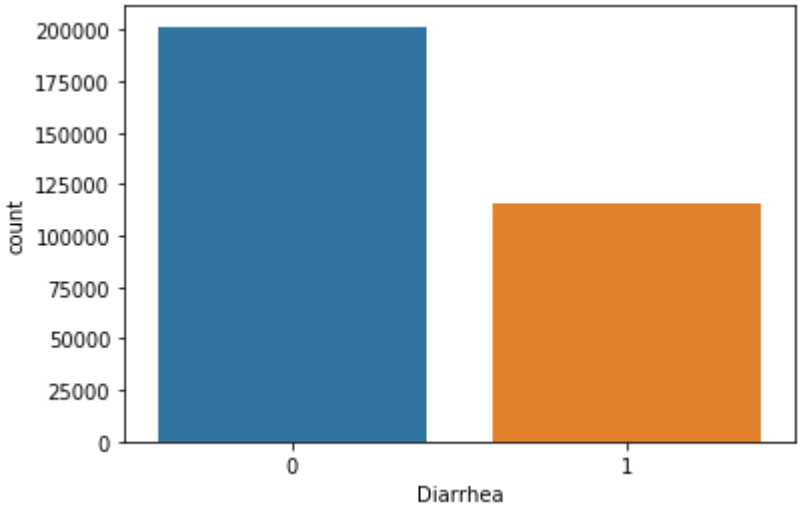


In [21]:

```
sns.countplot(data=df, x = 'Diarrhea')
```

Out[21]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1363e295188>
```

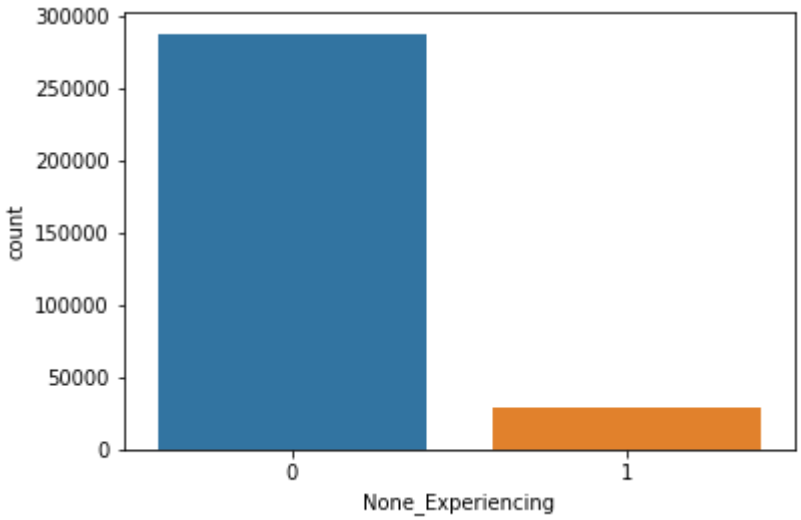


In [22]:

```
sns.countplot(data=df, x = 'None_Experiencing')
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1363d43b0c8>
```



Data Preprocessing

In [23]:

```
# check whether None_Sympton data are acceptable/logical
print(df[df.None_Sympton == 1].groupby('Fever').size())

print(df[df.None_Sympton == 1].groupby('Tiredness').size())

print(df[df.None_Sympton == 1].groupby('Dry-Cough').size())

print(df[df.None_Sympton == 1].groupby('Difficulty-in-Breathing').size())

print(df[df.None_Sympton == 1].groupby('Sore-Throat').size())
```

```
Fever
0    19800
dtype: int64
Tiredness
0    19800
dtype: int64
Dry-Cough
0    19800
dtype: int64
Difficulty-in-Breathing
0    19800
dtype: int64
Sore-Throat
0    19800
dtype: int64
```

In [24]:

```
# to check whether the patients data can have combination of symptoms
print(df[df.Fever == 1].groupby('Tiredness').size())
print(df[df.Fever == 1].groupby('Dry-Cough').size())
```

```
Tiredness
0    19800
1    79200
dtype: int64
Dry-Cough
0    39600
1    59400
dtype: int64
```

In [25]:

```
# check whether None_Experiencing data are acceptable/logical
print(df[df.None_Experiencing == 1].groupby('Nasal-Congestion').size())

print(df[df.None_Experiencing == 1].groupby('Runny-Nose').size())

print(df[df.None_Experiencing == 1].groupby('Diarrhea').size())
```

```
Nasal-Congestion
0    28800
dtype: int64
Runny-Nose
0    28800
dtype: int64
Diarrhea
0    28800
dtype: int64
```

In [26]:

```
# to check whether the patients data can have combination of experiences
print(df[df.Diarrhea == 1].groupby('Nasal-Congestion').size())
print(df[df.Diarrhea == 1].groupby('Runny-Nose').size())
```

```
Nasal-Congestion
0    57600
1    57600
dtype: int64
Runny-Nose
0    28800
1    86400
dtype: int64
```

In [27]:

```
# the contact data and country that have been visited are not related to the symptoms, henc

df.drop(['Contact_Dont-Know', 'Contact_Yes', 'Contact_No', 'Country'], axis=1, inplace=True
```

In [28]:

```
df
```

Out[28]:

	Fever	Tiredness	Dry-Cough	Difficulty-in-Breathing	Sore-Throat	None_Sympton	Pains	Nasal-Congestion	Runn Nos
0	1	1	1	1	1	0	1	1	
1	1	1	1	1	1	0	1	1	
2	1	1	1	1	1	0	1	1	
3	1	1	1	1	1	0	1	1	
4	1	1	1	1	1	0	1	1	
...	...	...	...	...	...	...	...	...	
316795	0	0	0	0	0	1	0	0	
316796	0	0	0	0	0	1	0	0	
316797	0	0	0	0	0	1	0	0	
316798	0	0	0	0	0	1	0	0	
316799	0	0	0	0	0	1	0	0	

316800 rows × 23 columns

In [29]:

```
# process the target columns

# collect the severity columns into the variable named severity_columns
severity_columns = df.filter(like='Severity_').columns

# change the value from 1 to its severity level while 0 to empty value ( for the next step
df['Severity_None'].replace({1:'None',0:''},inplace =True)
df['Severity_Mild'].replace({1:'Mild',0:''},inplace =True)
df['Severity_Moderate'].replace({1:'Moderate',0:''},inplace =True)
df['Severity_Severe'].replace({1:'Severe',0:''},inplace =True)

# gather the values from all the severity columns into 1 column named 'Severity'
df['Severity']=df[severity_columns].values.tolist()

# join all the values in the list and eliminate the ','
df['Severity'] = df['Severity'].apply(''.join)
```

In [30]:

```
df
```

Out[30]:

Diarrhea	...	Age_25-59	Age_60+	Gender_Female	Gender_Male	Gender_Transgender	Severity_Mild
1	...	0	0	0	1	0	Milc
1	...	0	0	0	1	0	Milc
1	...	0	0	0	1	0	Milc
1	...	0	0	0	1	0	
1	...	0	0	0	1	0	
...	...	...	...	...	...	...	..
0	...	0	1	0	0	1	
0	...	0	1	0	0	1	
0	...	0	1	0	0	1	
0	...	0	1	0	0	1	
0	...	0	1	0	0	1	

In [31]:

```
# encode the severity level data values into integers
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Severity'] = le.fit_transform(df['Severity'])
df
```

Out[31]:

Diarrhea	...	Age_25-59	Age_60+	Gender_Female	Gender_Male	Gender_Transgender	Severity_Mil
1	...	0	0	0	1	0	Mil
1	...	0	0	0	1	0	Mil
1	...	0	0	0	1	0	Mil
1	...	0	0	0	1	0	
1	...	0	0	0	1	0	
...	...	...	...	...	...	...	.
0	...	0	1	0	0	1	
0	...	0	1	0	0	1	
0	...	0	1	0	0	1	
0	...	0	1	0	0	1	
0	...	0	1	0	0	1	

In [32]:

```
# drop the severity columns and left only the one column named 'Severity'
df.drop(severity_columns, axis=1, inplace=True)
df
```

Out[32]:

Nasal-Congestion	Runny-Nose	Diarrhea	None_Experiencing	Age_0-9	Age_10-19	Age_20-24	Age_25-59	Age_60+
1	1	1	0	1	0	0	0	0
1	1	1	0	1	0	0	0	0
1	1	1	0	1	0	0	0	0
1	1	1	0	1	0	0	0	0
1	1	1	0	1	0	0	0	0
...	...	...	...	...	...	...	...	..
0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	0	0	1

try out one more time of EDA on processed data

In [33]:

```
# Calculate the correlation values
feature_cols = df.columns[:-1]
corr_values = df[feature_cols].corr()

# Simplify by emptying all the data below the diagonal
tril_index = np.tril_indices_from(corr_values)
print(tril_index)


# Make the unused values NaNs
for coord in zip(*tril_index):
    corr_values.iloc[coord[0], coord[1]] = np.NaN

# Stack the data and convert to a data frame
corr_values = (corr_values
               .stack()
               .to_frame()
               .reset_index()
               .rename(columns={'level_0':'feature1', 'level_1':'feature2', 0:'correlation'

# Get the absolute values for sorting
corr_values['abs_correlation'] = corr_values.correlation.abs()
```

```
(array([ 0,  1,  1,  2,  2,  2,  3,  3,  3,  3,  4,  4,  4,  4,  4,  5,  5,
         5,  5,  5,  5,  6,  6,  6,  6,  6,  6,  6,  7,  7,  7,  7,  7,  7,
         7,  7,  8,  8,  8,  8,  8,  8,  8,  8,  8,  9,  9,  9,  9,  9,  9,
         9,  9,  9,  9, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 11, 11,
        11, 11, 11, 11, 11, 11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 12,
        12, 12, 12, 12, 12, 12, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13,
        13, 13, 13, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
        14, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
        16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16,
        17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17,
        17, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18,
        18, 18, 18], dtype=int64), array([ 0,  0,  1,  0,  1,  2,  0,  1,  2,
3,  0,  1,  2,  3,  4,  0,  1,
         2,  3,  4,  5,  0,  1,  2,  3,  4,  5,  6,  0,  1,  2,  3,  4,  5,
         6,  7,  0,  1,  2,  3,  4,  5,  6,  7,  8,  0,  1,  2,  3,  4,  5,
         6,  7,  8,  9,  0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10,  0,  1,
         2,  3,  4,  5,  6,  7,  8,  9, 10, 11,  0,  1,  2,  3,  4,  5,  6,
         7,  8,  9, 10, 11, 12,  0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10,
        11, 12, 13,  0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13,
        14,  0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,
         0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
         0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17,  0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,
        16, 17, 18], dtype=int64))
```

In [34]:

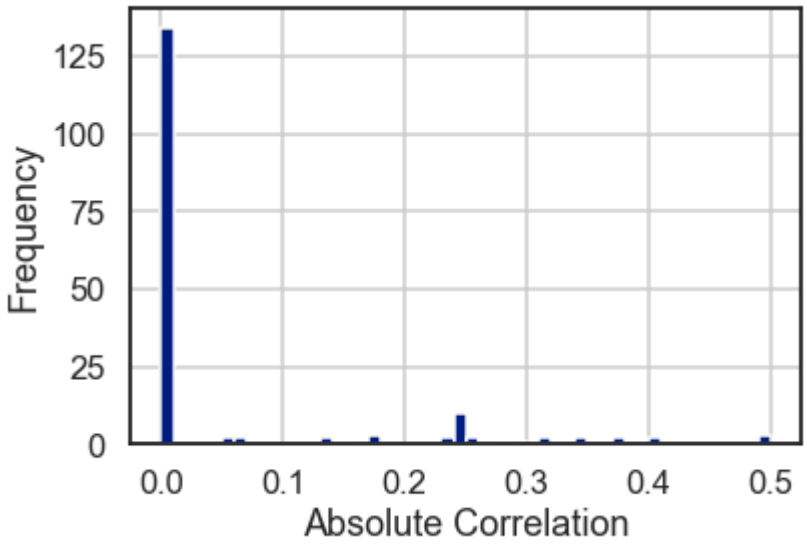
```
# plot out the frequency of absolute correlation values

sns.set_context('talk')
sns.set_style('white')
sns.set_palette('dark')

ax = corr_values.abs_correlation.hist(bins=50)
ax.set(xlabel='Absolute Correlation', ylabel='Frequency')
```

Out[34]:

```
[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Absolute Correlation')]
```



In [35]:

```
# Find the most highly correlated values
# corr_values.sort_values('correlation', ascending=False).query('abs_correlation>0.8')
```

Build predictive model

In [36]:

```
# prepare training set and testing set

# features stored in x variable, target/label stored in y variable
x = df.copy()
y = df.Severity
del x['Severity']
```

In [37]:

```
# check on the first 5 rows from x variable
x.head()
```

Out[37]:

	Fever	Tiredness	Dry-Cough	Difficulty-in-Breathing	Sore-Throat	None_Sympton	Pains	Nasal-Congestion	Runny-Nose	Di
0	1	1	1	1	1	0	1	1	1	
1	1	1	1	1	1	0	1	1	1	
2	1	1	1	1	1	0	1	1	1	
3	1	1	1	1	1	0	1	1	1	
4	1	1	1	1	1	0	1	1	1	

In [39]:

```
# check on the first 5 rows from y variable
y.head()
```

Out[39]:

```
0    0
1    0
2    0
3    1
4    1
Name: Severity, dtype: int32
```

In [40]:

```
# splitting the x and y into training set and testing set

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

K-Nearest Neighbours (KNN)

In [41]:

```
# import libraries
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

In [42]:

```
# build the default KNN model and fit the training set into it
result = {}
for i in range(10):
    k = i+1
    knn = KNeighborsClassifier(n_neighbors=k)
    knn = knn.fit(x_train, y_train)
    y_pred = knn.predict(x_test)
    accuracy = accuracy_score(y_test, y_pred)
    result[k] = accuracy

print(result)
```

```
{1: 0.24348695286195285, 2: 0.2436763468013468, 3: 0.24058291245791247, 4:
0.23929924242424241, 5: 0.2383627946127946, 6: 0.23731060606060606, 7: 0.236
50042087542086, 8: 0.23525883838383838, 9: 0.23477483164983165, 10: 0.234448
6531986532}
```

In [44]:

```
# build another KNN model with tuned parameter : weights=uniform, distance=Manhattan
result_uni_1 = {}
for i in range(10):
    k = i+1
    knn = KNeighborsClassifier(n_neighbors=k, p=1)
    knn = knn.fit(x_train, y_train)
    y_pred = knn.predict(x_test)
    accuracy = accuracy_score(y_test, y_pred)
    result_uni_1[k] = accuracy

print(result_uni_1)
```

```
{1: 0.24348695286195285, 2: 0.2436763468013468, 3: 0.24058291245791247, 4:
0.23929924242424241, 5: 0.2383627946127946, 6: 0.23731060606060606, 7: 0.236
50042087542086, 8: 0.23525883838383838, 9: 0.23477483164983165, 10: 0.234448
6531986532}
```

In [45]:

```
# build another KNN model with tuned parameter : weights=distance, distance=Manhattan
result_dist_1 = {}
for i in range(10):
    k = i+1
    knn = KNeighborsClassifier(n_neighbors=k, weights='distance', p=1)
    knn = knn.fit(x_train, y_train)
    y_pred = knn.predict(x_test)
    accuracy = accuracy_score(y_test, y_pred)
    result_dist_1[k] = accuracy

print(result_dist_1)
```

```
{1: 0.24348695286195285, 2: 0.2436763468013468, 3: 0.24058291245791247, 4:
0.23929924242424241, 5: 0.2383627946127946, 6: 0.23731060606060606, 7: 0.236
50042087542086, 8: 0.23525883838383838, 9: 0.23477483164983165, 10: 0.234448
6531986532}
```

In [46]:

```
# build another KNN model with tuned parameter : weights=distance, distance=Euclidean
result_dist_2 = {}
for i in range(10):
    k = i+1
    knn = KNeighborsClassifier(n_neighbors=k, weights='distance', p=2)
    knn = knn.fit(x_train, y_train)
    y_pred = knn.predict(x_test)
    accuracy = accuracy_score(y_test, y_pred)
    result_dist_2[k] = accuracy

print(result_dist_2)
```

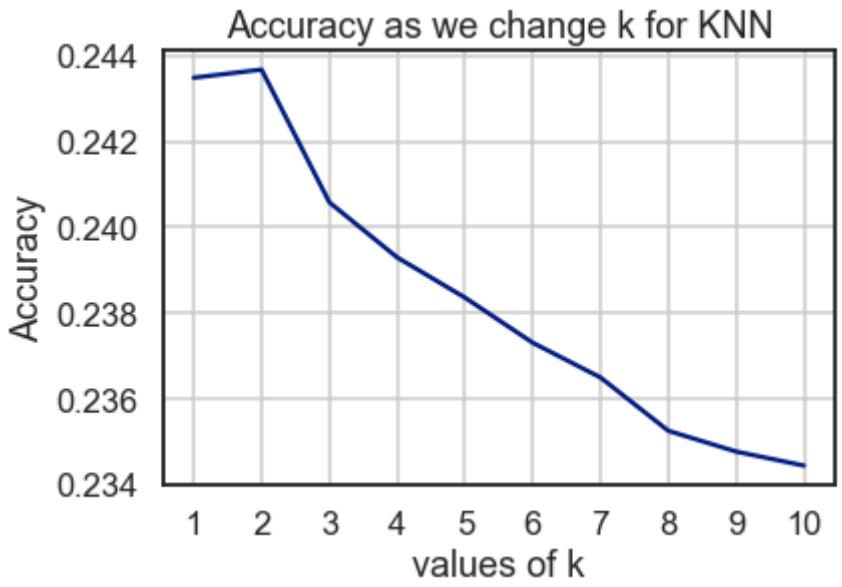
```
{1: 0.24348695286195285, 2: 0.2436763468013468, 3: 0.24058291245791247, 4:
0.23929924242424241, 5: 0.2383627946127946, 6: 0.23731060606060606, 7: 0.236
50042087542086, 8: 0.23525883838383838, 9: 0.23477483164983165, 10: 0.234448
6531986532}
```

In [47]:

```
# plot the accuracy vs k value, to find out the best k for building a KNN model
plt.plot([i+1 for i in range(10)], list(result.values()))
plt.xlabel('values of k')
plt.ylabel('Accuracy')
plt.title('Accuracy as we change k for KNN')
plt.grid(True)
plt.xticks([i+1 for i in range(10)])
```

Out[47]:

```
([<matplotlib.axis.XTick at 0x1363f76a188>,
  <matplotlib.axis.XTick at 0x1363f7657c8>,
  <matplotlib.axis.XTick at 0x1363f765388>,
  <matplotlib.axis.XTick at 0x1363f78ac08>,
  <matplotlib.axis.XTick at 0x1363f78d3c8>,
  <matplotlib.axis.XTick at 0x1363f78dc88>,
  <matplotlib.axis.XTick at 0x1363f791608>,
  <matplotlib.axis.XTick at 0x1363f795208>,
  <matplotlib.axis.XTick at 0x1363f795848>,
  <matplotlib.axis.XTick at 0x1363f791448>],
 <a list of 10 Text xticklabel objects>)
```



In [48]:

```
# cross validation
from sklearn.model_selection import cross_val_score

# select the best k to be checked for cross validation
knn = KNeighborsClassifier(n_neighbors=2)

cross_val = cross_val_score(knn, x_train, y_train, cv=4)
cross_val
```

Out[48]:

```
array([0.2412518 , 0.24457071, 0.24516595, 0.24186508])
```

In [50]:

```
# evaluate the KNN model performance using different measurements
from sklearn.metrics import precision_recall_fscore_support as score

knn = knn.fit(x_train, y_train)
y_pred = knn.predict(x_test)

precision, recall, fscore, _ = score(y_test, y_pred, average='weighted')
print("precision=", precision)
print("recall=", recall)
print("fscore=", fscore)
```

```
precision= 0.24135746919702217
recall= 0.2436763468013468
fscore= 0.22151733373763083
```

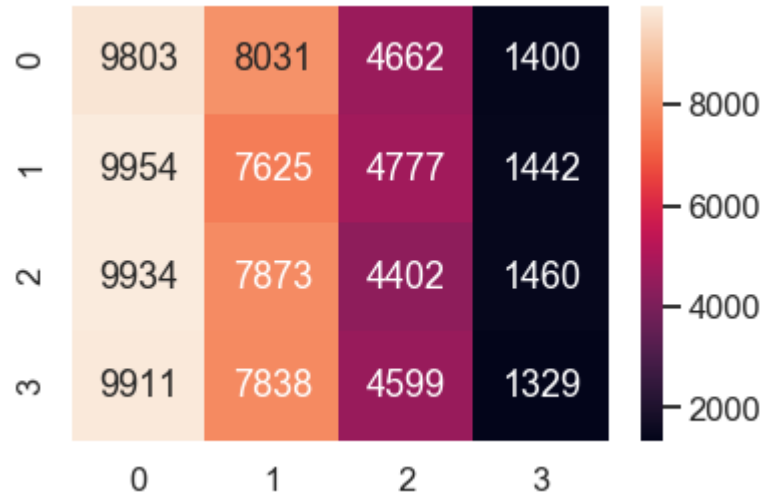
In [51]:

```
# confusion matrix of the KNN model
from sklearn.metrics import confusion_matrix

cm_knn = confusion_matrix(y_test, y_pred)
sns.heatmap(cm_knn, annot=True, fmt='d')
ax.set(title='K-Nearest Neighbors, k=2')
```

Out[51]:

```
[Text(0.5, 1, 'K-Nearest Neighbors, k=2')]
```



Logistic Regression

In [52]:

```
# import libraries
from sklearn.model_selection import StratifiedShuffleSplit

# Get the split indexes
strat_shuf_split = StratifiedShuffleSplit(n_splits=1,
                                          test_size=0.3,
                                          random_state=42)
train_idx, test_idx = next(strat_shuf_split.split(df[feature_cols], df.Severity))

# Create the dataframes
x_train = df.loc[train_idx, feature_cols]
y_train = df.loc[train_idx, 'Severity']
x_test = df.loc[test_idx, feature_cols]
y_test = df.loc[test_idx, 'Severity']
```

In [53]:

```
# have a look on the distribution of each values on y_train
y_train.value_counts(normalize=True)
```

Out[53]:

```
3    0.25
2    0.25
1    0.25
0    0.25
Name: Severity, dtype: float64
```

In [54]:

```
# import libraries
from sklearn.linear_model import LogisticRegression

# Ignore some warnings which are sure to show up in this question
from warnings import simplefilter
from sklearn.exceptions import ConvergenceWarning
simplefilter("ignore", category=ConvergenceWarning)
simplefilter("ignore", category=FutureWarning)
```

In [55]:

```
# Standard logistic regression
lr = LogisticRegression().fit(x_train, y_train)

from sklearn.linear_model import LogisticRegressionCV

# L1 regularized logistic regression with parameter multi-class='ovr'
lr_l1_ovr = LogisticRegressionCV(Cs=10, cv=4, penalty='l1', solver='liblinear', multi_class

# L2 regularized logistic regression with parameter multi-class='ovr'
lr_l2_ovr= LogisticRegressionCV(Cs=10, cv=4, penalty='l2', multi_class='ovr').fit(x_train,

# L1 regularized logistic regression with parameter multi-class='multinomial'
lr_l1_mn = LogisticRegressionCV(Cs=10, cv=4, penalty='l1', solver='saga', multi_class='mult

# L2 regularized logistic regression with parameter multi-class='multinomial'
lr_l2_mn= LogisticRegressionCV(Cs=10, cv=4, penalty='l2', multi_class='multinomial').fit(x_
```

In [56]:

```
# Predict the class and the probability for each parameter-tuned logistic regression models
y_pred = list()
y_prob = list()
coeff_labels = ['lr', 'l1_ovr', 'l2_ovr', 'l1_mn', 'l2_mn']
coeff_models = [lr, lr_l1_ovr, lr_l2_ovr, lr_l1_mn, lr_l2_mn]
for lab,mod in zip(coeff_labels, coeff_models):
    y_pred.append(pd.Series(mod.predict(x_test), name=lab))
    y_prob.append(pd.Series(mod.predict_proba(x_test).max(axis=1), name=lab))

y_pred = pd.concat(y_pred, axis=1)
y_prob = pd.concat(y_prob, axis=1)

# get the first 5 predicted values
y_pred.head()
```

Out[56]:

	lr	l1_ovr	l2_ovr	l1_mn	l2_mn
0	3	0	3	3	3
1	0	0	0	3	0
2	0	0	1	3	1
3	0	0	0	3	0
4	3	0	3	3	3

In [57]:

```
# get the calculated probability on getting the first 5 predicted values
y_prob.head()
```

Out[57]:

	lr	l1_ovr	l2_ovr	l1_mn	l2_mn
0	0.252850	0.250004	0.250955	0.25052	0.251133
1	0.253502	0.250004	0.251267	0.25052	0.251509
2	0.251290	0.250004	0.250676	0.25052	0.250755
3	0.252879	0.250004	0.251536	0.25052	0.251760
4	0.251678	0.250004	0.250473	0.25052	0.250572

In [58]:

```
# import libraries used for evaluating the logistic regression models performance
from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score
from sklearn.preprocessing import label_binarize
```

In [59]:

```
# evaluates the logistic regression models performance using different measurements
metrics = list()
cm = dict()
for lab in coeff_labels:
    # Preciision, recall, f-score from the multi-class support function
    precision, recall, fscore, _ = score(y_test, y_pred[lab],average='weighted')
    # The usual way to calculate accuracy
    accuracy = accuracy_score(y_test, y_pred[lab])
    # ROC-AUC scores can be calculated by binarizing the data
    auc = roc_auc_score(label_binarize(y_test, classes=[0,1,2,3]),
                        label_binarize(y_pred[lab], classes=[0,1,2,3]),
                        average='weighted')

    # Last, the confusion matrix
    cm[lab] = confusion_matrix(y_test, y_pred[lab])
    metrics.append(pd.Series({'precision':precision, 'recall':recall,
                              'fscore':fscore, 'accuracy':accuracy,
                              'auc':auc},
                             name=lab))

metrics = pd.concat(metrics, axis=1)
```

```
C:\Users\MichelleLin\anaconda3\lib\site-packages\sklearn\metrics\_classifica
tion.py:1272: UndefinedMetricWarning: Precision and F-score are ill-defined
and being set to 0.0 in labels with no predicted samples. Use `zero_division
` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\MichelleLin\anaconda3\lib\site-packages\sklearn\metrics\_classifica
tion.py:1272: UndefinedMetricWarning: Precision and F-score are ill-defined
and being set to 0.0 in labels with no predicted samples. Use `zero_division
` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [60]:

```
#Run the metrics to look at the precision, recall, fscore, accuracy and auc respectively
metrics
```

Out[60]:

	lr	l1_ovr	l2_ovr	l1_mn	l2_mn
precision	0.244944	0.0625	0.244194	0.0625	0.244548
recall	0.244960	0.2500	0.244213	0.2500	0.244592
fscore	0.243777	0.1000	0.243786	0.1000	0.244142
accuracy	0.244960	0.2500	0.244213	0.2500	0.244592
auc	0.496640	0.5000	0.496142	0.5000	0.496395

In [61]:

```
# plot out all the confusion matrix respectively
fig, axList = plt.subplots(nrows=3, ncols=2)
axList = axList.flatten()
fig.set_size_inches(12, 10)

axList[-1].axis('off')

for ax,lab in zip(axList[:-1], coeff_labels):
    sns.heatmap(cm[lab], ax=ax, annot=True, fmt='d');
    ax.set(title=lab);

plt.tight_layout()
```