

PREDICTION AND CLASSIFICATION OF COVID-19 DEATH AND RECOVERED CASES USING ML

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

By

THIRSHA ANGELINE. R (REG NO: 39290112)

KALAISELVI . S (REG NO: 39290045)



DEPARTMENT OF COMPUTER SCIENCE

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI –
600119, TAMILNADU

MARCH 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY) Accredited with Grade "A"
by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI- 600119
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Thirshaangeline. R (reg no:39290112) and kalaiselvi. S (reg no: 39290045)** carried out the project entitled " **PREDICTION AND CLASSIFICATION OF COVID -19 DEATH AND RECOVERED CASES USING ML**" under my supervision from November 2021 to march 2022

Internal Guide **Dr. Dr. Jancy**

Head of the Department **Dr. Lakshmanan**

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **THIRSHA ANGELINE (reg no: 39290112)** hereby declare that the Project Report entitled

"PREDICTION AND CLASSIFICATION OF COVID-19 DEATH AND RECOVERED CASES USING ML me under the guidance of **Dr.JANCY**
I submitted in partial fulfillment of the requirements for the award of Bachelor of Science degree in Computer Science.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

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

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

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. JANCY** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

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

ChapterNO	TITLE
.	ABSTRACT v
	LIST OF FIGURES viii
1	.INTRODUCTION
1.1	OVERVIEW OF PROJECT 1
2	LITERATURE SURVEY
3	AIM AND SCOPE OF PRESENT INVESTIGATION
3.1	AIM OF THE PROJECT
3.2	SCOPE AND OBJECTIVE 5
3.3	SYSTEM REQUIREMENTS 6
3.3.1	HARDWARE REQUIREMENTS 6 3
3.3.2	SOFTWARE REQUIREMENTS 6
3.4	SOFTWARE USED 7
3.4.1	PYTHON LANGUAGE 7
3.4.2	PYTHON CHARACTERISTICS 7
3.4.3	APPLICATIONS OF PYTHON 7
3.4.4	OPENCV PACKAGE 8
3.5	ANACONDA NAVIGATOR 9
3.5.1	ANACONDA 9
3.5.2	APPLICATIONS IN ANACONDA 11
3.5.3	VS CODE 12 3.
5.4	NEW FEATURES OF ANACONDA 5.
4	EXPERIMENTAL OR MATERIAL METHODS
4.1	DESIGN METHODOLOGY 14

4.1.1 EXISTING SYSTEM 14

4.1.2 PROPOSED SYSTEM 14

4.2 MODULE DESCRIPTION 14

4.3 ARCHITECTURE DIAGRAM

4.4 MODEL TRAINING

5.CONCLUSION AND FUTURE ENHANCEMENT

5.1 CONCLUSION

5.2 FUTURE ENHANCEMENT

REFERENCES

APPENDIX

SOURCE CODE

CHAPTER 11

INTRODUCTION

OVERVIEW OF PROJECT

Over the last decade, machine learning (ML) proved to be an important study field, addressing many incredibly complicated and elaborate problems in the real world. [Nearly all real-world areas, including healthcare, autonomous vehicles (AV) , Enterprises software, NLP, smart robotics, sports, climatic simulation, voicing and image processing. were included in the application fields.

Important ML areas are expected. Many standard ML algorithms were used to direct potential outcomes in many application fields, including temperature, disease forecasting, stock market prediction, and disease prediction. Different models of regression and neural networks are broadly applicable to forecasts of future disease conditions in patients [5]. Diverse experiments were undertaken to predict various ML disorders, such as cardiovascular disease, and coronary heart disease. And the research focuses in particular on living COVID-19 predictions and the study Focuses even on the prediction and early response of COVID-19 outbreaks To monitor the current situation by decision making through the prediction process in order to direct earlier interventions in the efficient management of the disease.

Important ML areas are expected. Many standard ML algorithms were used to direct potential outcomes in many application fields, including temperature, disease forecasting, stock market prediction, and disease prediction. Different models of regression and neural networks are broadly applicable to forecasts of future disease conditions in patients [5]. Diverse experiments were undertaken to predict various ML disorders, such as cardiovascular disease, and coronary heart disease. And the research focuses in particular on living COVID-19 predictions and the study Focuses even on the prediction and early response of COVID-19 outbreaks To monitor the current situation by decision making through the prediction process in order to direct earlier interventions in the efficient management of the disease.

LITRATUR REVIEWE:

S.NO:	AUTHOUR	TITTLE	YEAR	ALGORITHM	DRAWBACK
1	S. Fu, C. C. Wyles, D. R. Osmon, M. L. Carvour, E. Sagheb, T. Ramazanian, W. K. Kremers, D. G. Lewallen, D. J. Berry, S. Sohn, and H. M. KremersD. Mazzei, F. Chiarello, and G. Fantoni	Automated detection of periprosthetic joint infections and data elements using natural language processing,	2021	Conversational user interface	After that, NLP is introduced to process these data
2	D. Mazzei, F. Chiarello, and G. Fantoni.	Analyzing social robotics research with natural language processing techniques	2021	Natural language processing	carrying out a text-driven paper filtering and classification method
3	A. M. C. H. Attanayake, S. S. N. Perera, and S. Jayasinghe	Phenomenological modelling of COVID-19 epidemics in Sri Lanka, Italy and Hebei Province of China	2020	crawler	the curves are similar to the logistic growth model
4	D. Shung, C. Tsay, L. Laine, D. Chang, F. Li, P. Thomas, C. Partridge, M. Simonov, A. Hsiao, J. K. Tay, and A. Taylor	Early identification of patients with acute gastrointestinal bleeding using natural language processing and decision rules	2020	regression	The change of the number of positive words or negative word

3.3 3.3 SYSTEM REQUIREMENTS

3.3.1 3.3.1 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. The minimal hardware requirements are as follows

- , 1. OS : Windows7,8 and 10(32 and 64bit)_
- 2. RAM : 4GB

: 3.3.2 3.3.2 Software Requirements

Software requirements deals with defining resource requirements and prerequisites that needs to be installed on a computer to provide functioning of an application. The minimal software requirements are as follows

- : python :
- 2 : anaconda

3.4 3.4 SOFTWARE USED:

3.4.1 Python Language

Python is an object-oriented programming language created by Guido Rossum in 1989. It is ideally designed for rapid prototyping of complex applications. It has interfaces to many OS system calls and libraries and is extensible to C or C++. Many large companies use the Python programming language include NASA, Google, YouTube, BitTorrent, etc. Python programming is widely used in Artificial Intelligence, Natural Language Generation, Neural Networks and other advanced fields of Computer Science. Python had deep focus on code readability & this class will teach you python from basics.

3.4.2 Python Characteristics •

It provides rich data types and easier to read syntax than any other programming languages •

It is a platform independent scripted language with full access to operating system API's

- Compared to other programming languages, it allows more run-time flexibility
- It includes the basic text manipulation facilities of Perl and Awk
- A module in Python may have one or more classes and free functions
- Libraries in Python are cross-platform compatible with Linux, Macintosh, and Windows
- For building large applications, Python can be compiled to byte-code
- Python supports functional and structured programming as well as OOP
- It supports interactive mode that allows interacting Testing and debugging of snippets of code
- In Python, since there is no compilation step, editing, debugging and testing is fast.

3.4.3 Applications of Python Programming

Web Applications:

You can create scalable Web Apps using frameworks and CMS (Content Management System) that are built on Python. Some of the popular platforms for creating Web Apps are: Django, Flask, Pyramid, Plone, Django CMS. Sites like Mozilla, Reddit, Instagram and PBS are written in Python.

Scientific and Numeric Computing:

There are numerous libraries available in Python for scientific and numeric computing. There are libraries like: SciPy and NumPy that are used in general purpose computing. And, there are specific libraries like: Earthy for earth science, Astray for Astronomy and so on. Also, the language is heavily used in machine learning, data mining and deep learning.

Creating software Prototypes:

Python is slow compared to compiled languages like C++ and Java. It might not be a good choice if resources are limited and efficiency is a must. However, Python is

a great language for creating prototypes. For example: You can use Pygmy (library for creating games) to create your game's prototype first. If you like the prototype, you can use language like C++ to create the actual game.

Good Language to Teach Programming

: Python is used by many companies to teach programming to kids and newbies. It is a good language with a lot of features and capabilities. Yet, it's one of the easiest languages to learn because of its simple easy-to-use syntax.

3.4.4 OpenCV Package

Python is a general-purpose programming language started by Guido van Rossum, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability

Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python 9 wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation.

And the support of NumPy makes the task easier. NumPy is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from NumPy arrays. So whatever operations you can do in NumPy, you can combine it with OpenCV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this

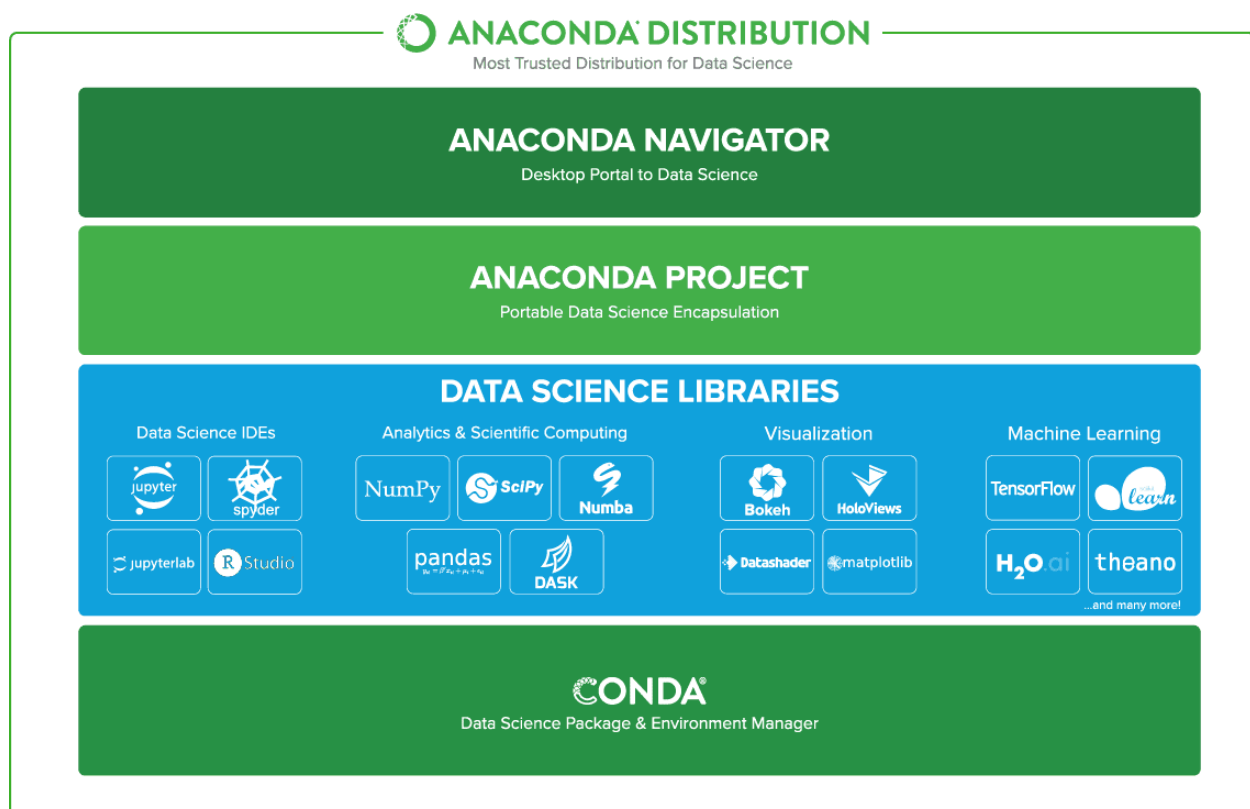
. So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

3.5 ANACONDA NAVIGATO

3.5.1 Anaconda:

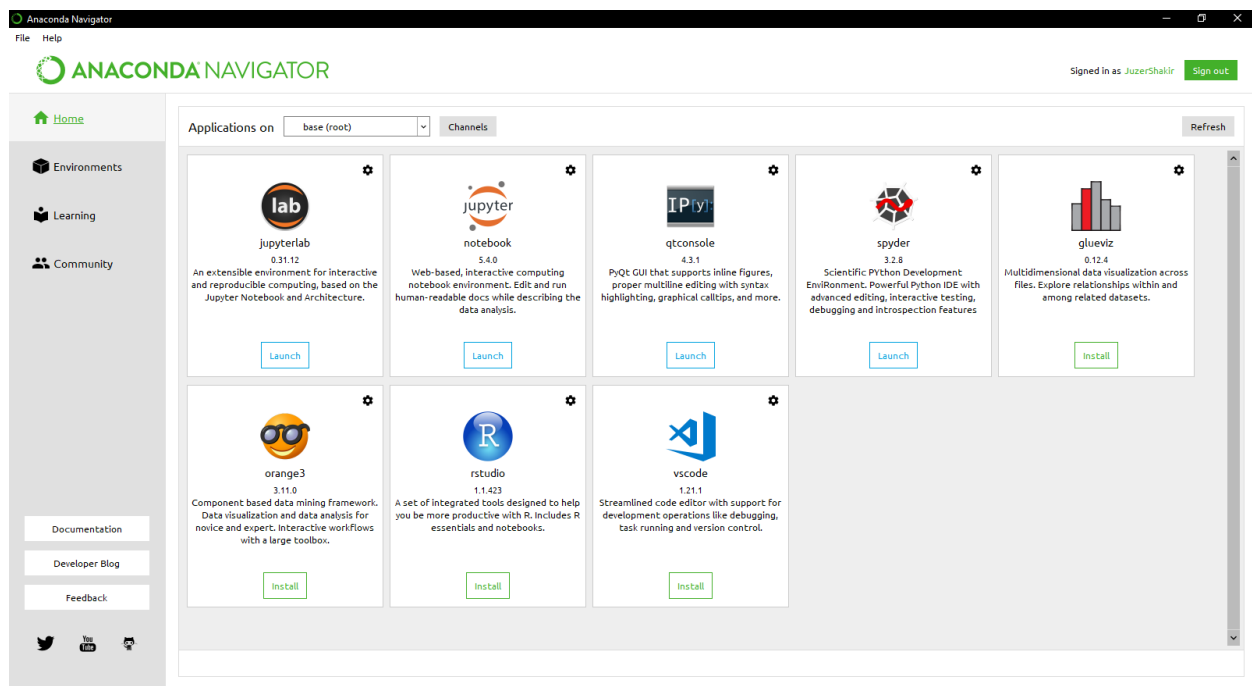
Anaconda is a free and open source, easy to install distribution of Python and R programming languages. Anaconda provides a working environment which is used for scientific computing, data science, statistical analysis and machine learning.

The latest distribution of Anaconda is Anaconda 5.3 and is released in October, 2018. It has the conda package, environment manager and a collection of 1000+ open source packages long with free community support.



What is Anaconda Navigator?

Anaconda Navigator is a desktop graphical user interface (GUI) included in the Anaconda distribution. It allows us to launch applications provided in the Anaconda distribution and easily manage conda packages, environments and channels without the use of command-line commands. It is available for Windows, macOS and Linux.



Applications Of Anaconda

- The Anaconda distribution comes with the following applications along with Anaconda Navigator
- . 1. JupyterLab
- 2. Jupyter Notebook
- 3. Qt Console
- 4. Spyder
- 5. Glueviz
- 6. Orange3
- 7. RStudio 12
- 8. Visual Studio Code

3.5.3 VS Code:

VS Code is a free, open source streamlined cross-platform code editor with excellent support for Python code editing, IntelliSense, debugging, linting, version control, and more. Additionally, the Python Extension for Visual Studio Code tailors VS Code into a Python IDE. It is a streamlined code editor with support for development operations like debugging, task running and version control

. VS Code is free for both private and commercial use, runs on Windows, macOS, and Linux, and includes support for linting, debugging, task running, version control and Git integration, IntelliSense code completion, and conda environments. VS Code is openly extensible and many extensions are available.

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.

3.5.4 New Features of Anaconda 5.3

Compiled with Latest Python release: Anaconda 5.3 is compiled with Python 3.7, taking advantage of Python's speed and feature improvements.

- **Better Reliability:**

The reliability of Anaconda has been improved in the latest release by capturing and storing the package metadata for installed packages

- **Enhanced CPU Performance**

: The Intel Math Kernel Library 2019 for Deep Neural Networks(MKL 2019) has been introduced in Anaconda 5.3 distribution. Users deploying TensorFlow can make use of MKL 2019 for Deep Neural Networks. These Python binary packages are provided to achieve high CPU performance

- **New packages are added:**

There are over 230 packages which has been updated and added in the new release

. • Work in Progress:

There is a casting bug in Numpy with Python 3.7 but the team is currently working on patching it until Numpy is update

CHAPTER 4 4.

EXPERIMENTAL OR MATERIAL METHODS

4.1 DESIGN METHODOLOGY

4.1.1 Existing System:

Typically, the existing mechanisms assumed that the accuracy of prediction was achieved. But this wasn't the case then, hence, it must be improved  further to increase the classification accuracy. Also, other research works addressed these issues by introducing efficient combination. Existing Models based on feature selection and classification raised some issues regarding with training dataset and Test dataset

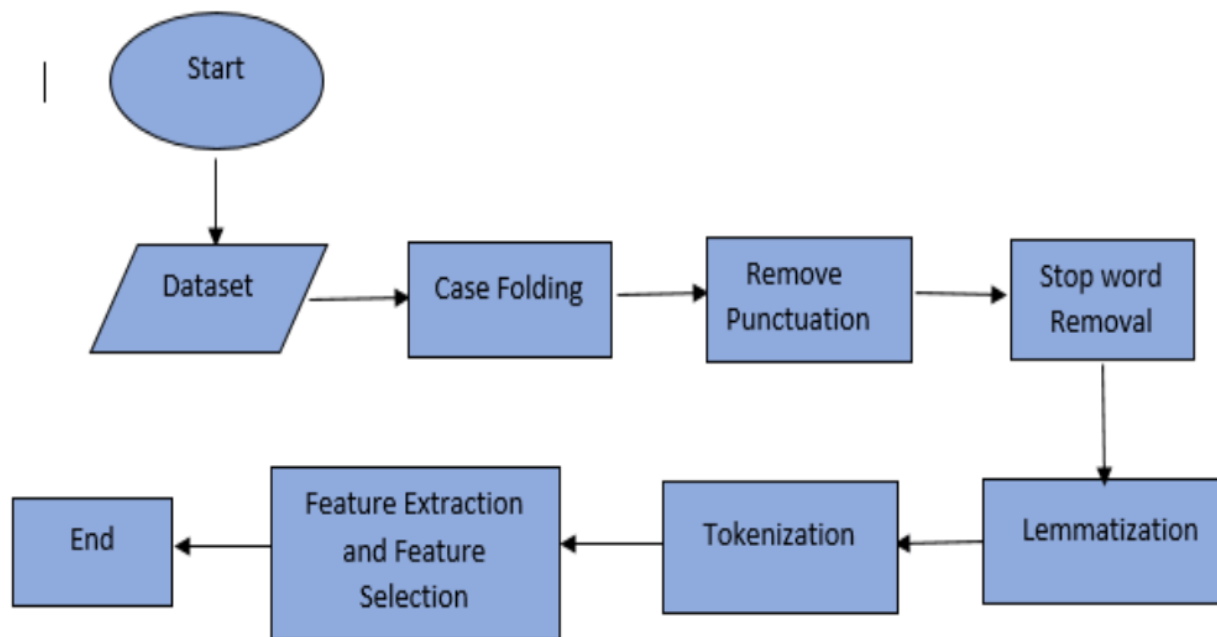
. Dis-Advantage: ?

- Certain approaches being applicable only for small data. ?
- Certain combination of classifier over fit with data set while others are under fit. ?
- Some approaches are not adoptable for real time collection of database implementation.

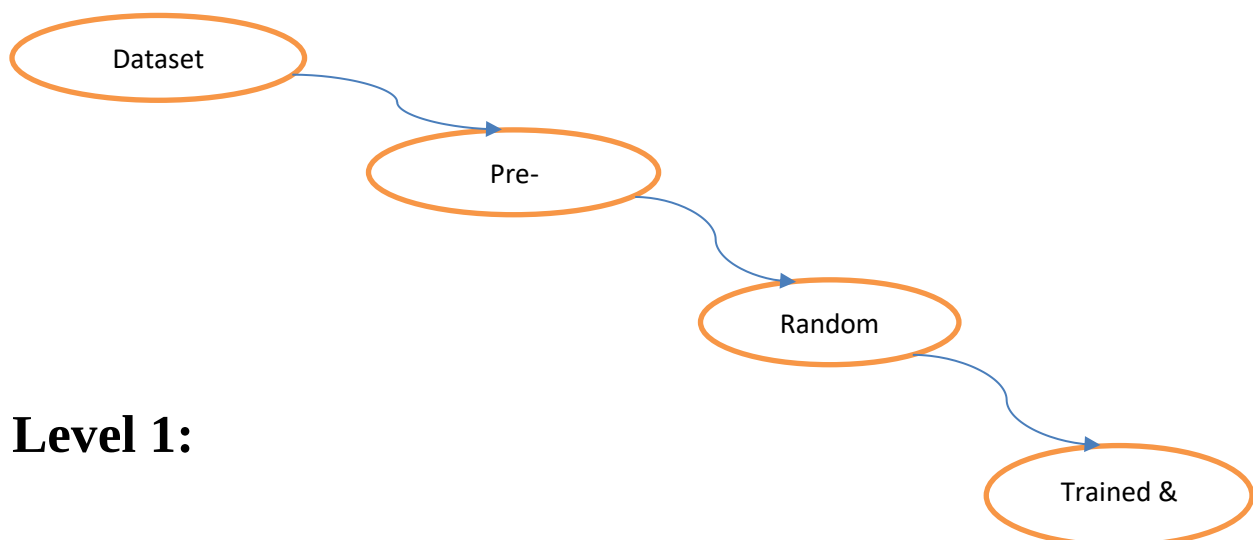
4.1.2 4.1.2 Proposed System:

This model is proposed with an algorithm to predict the same. Data pertaining to the study were collected from Kaggle for which the covid-19 analysis and prediction is done, and also suitable data pre-processing methods were applied. This proposed model is also compared with other traditional classification algorithms such as Existing techniques, SVR and Random forest and XGBOOST with respect to accuracy, precision and recall.

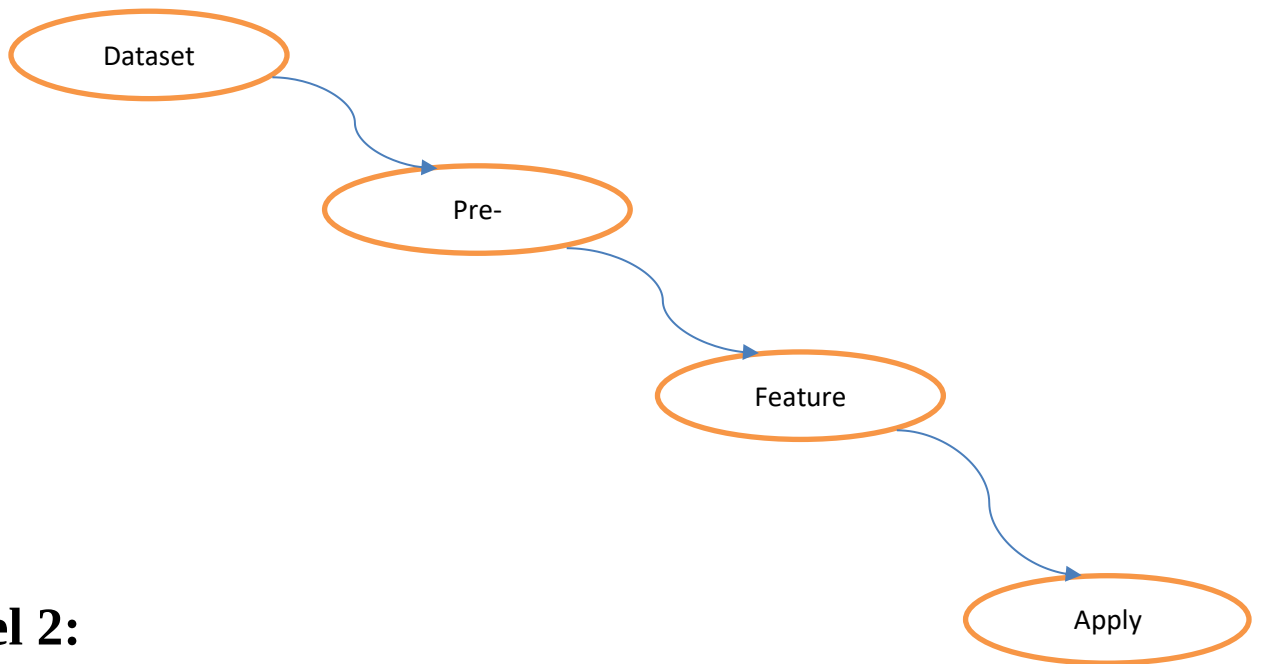
System Architecture:



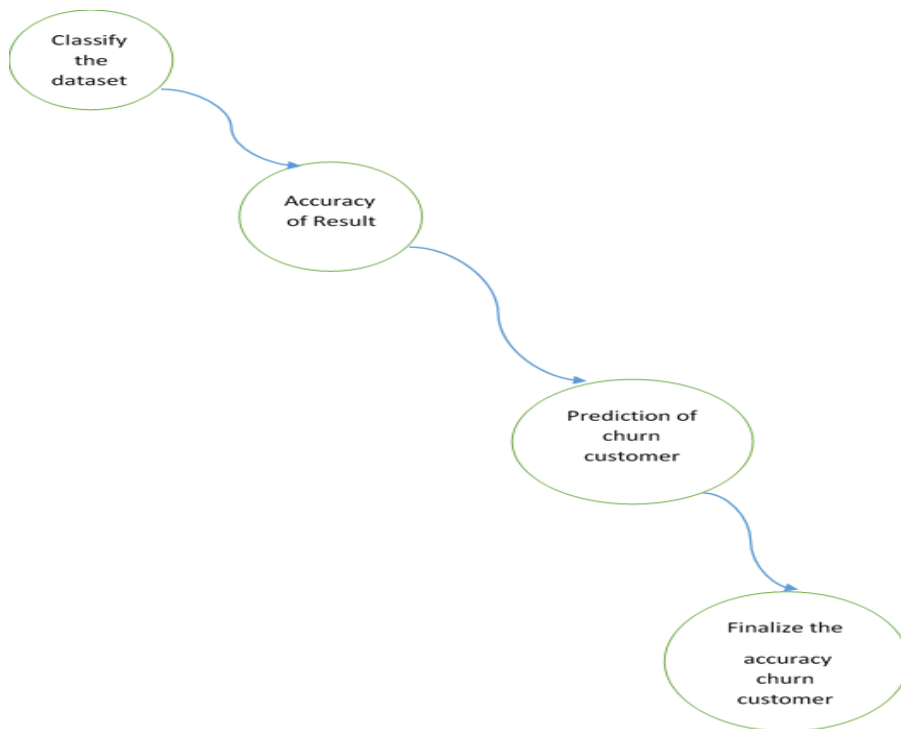
Data flow Diagram



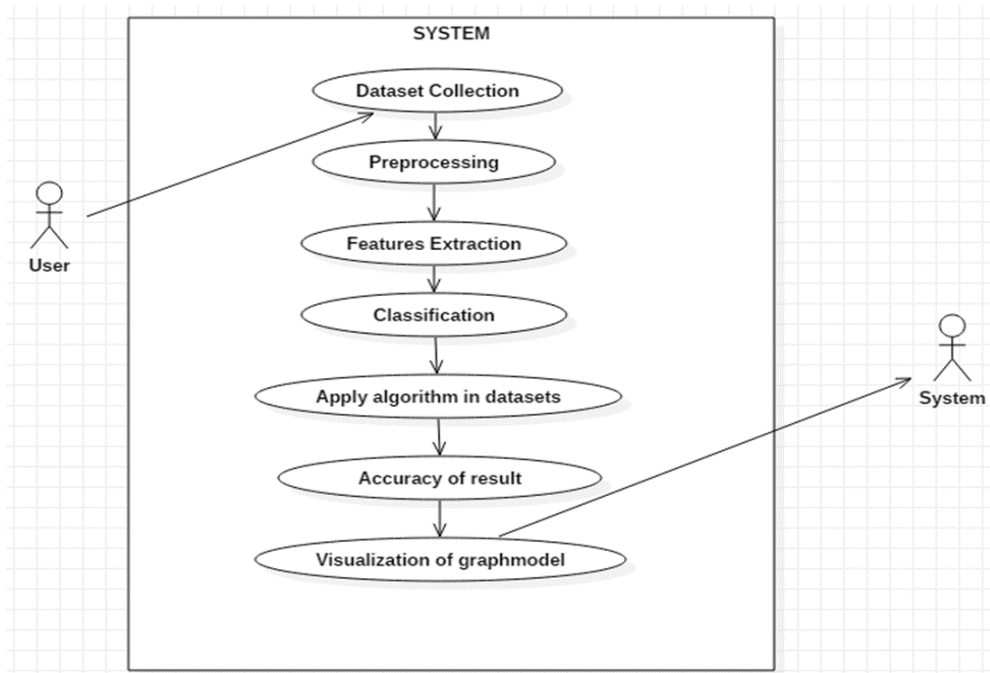
Level 1:



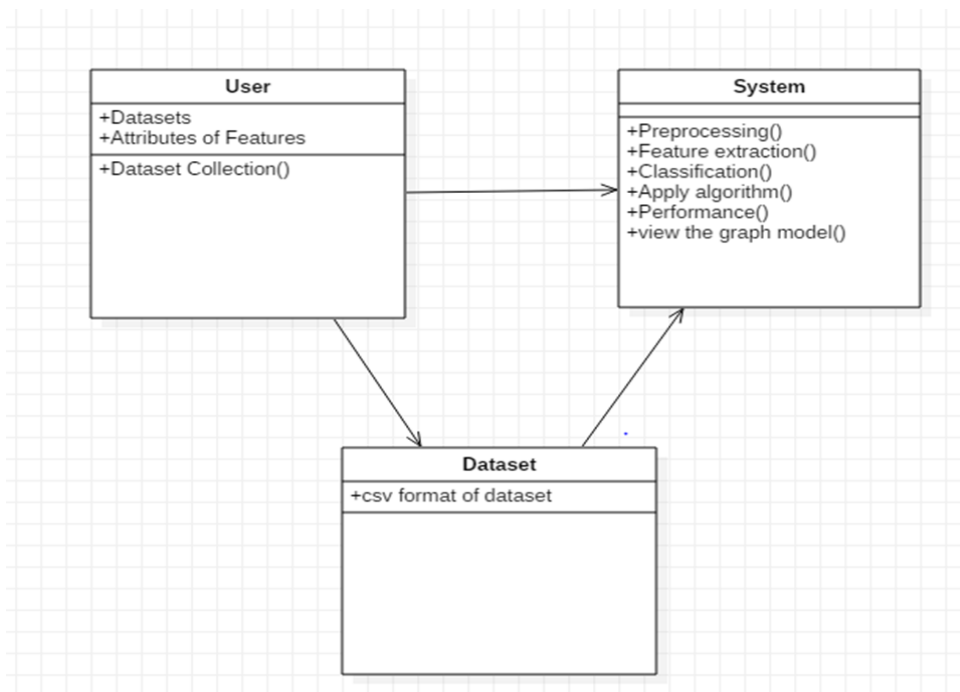
Level 2:



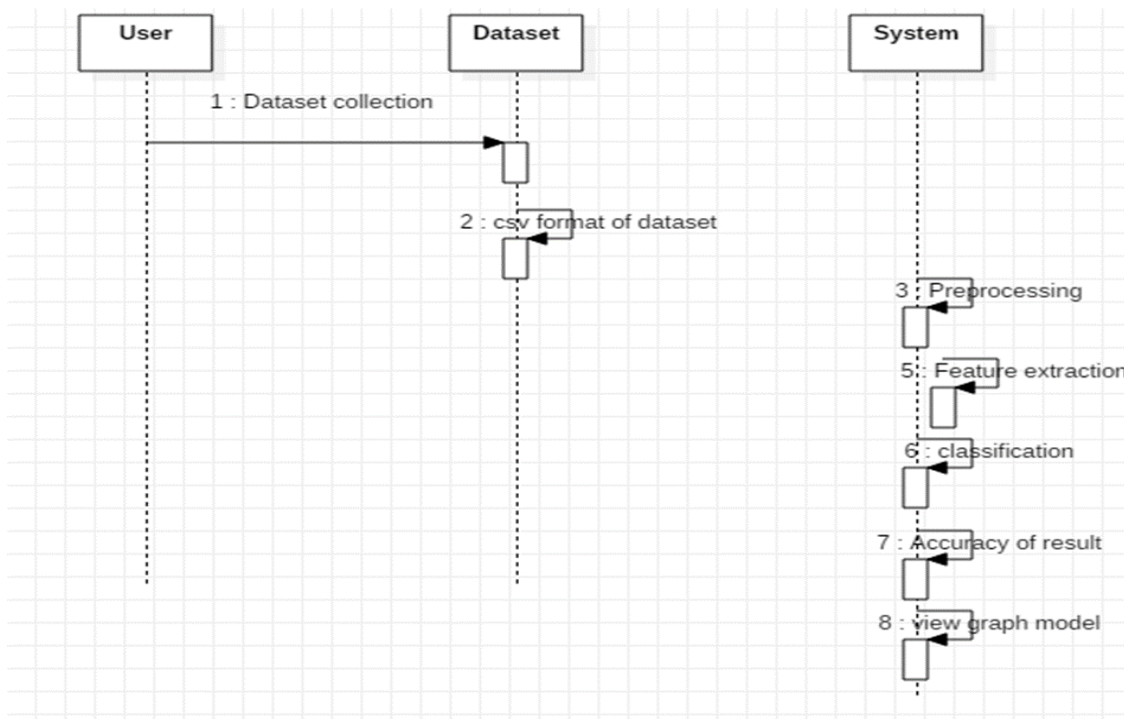
Uml Diagram Use Case Diagram



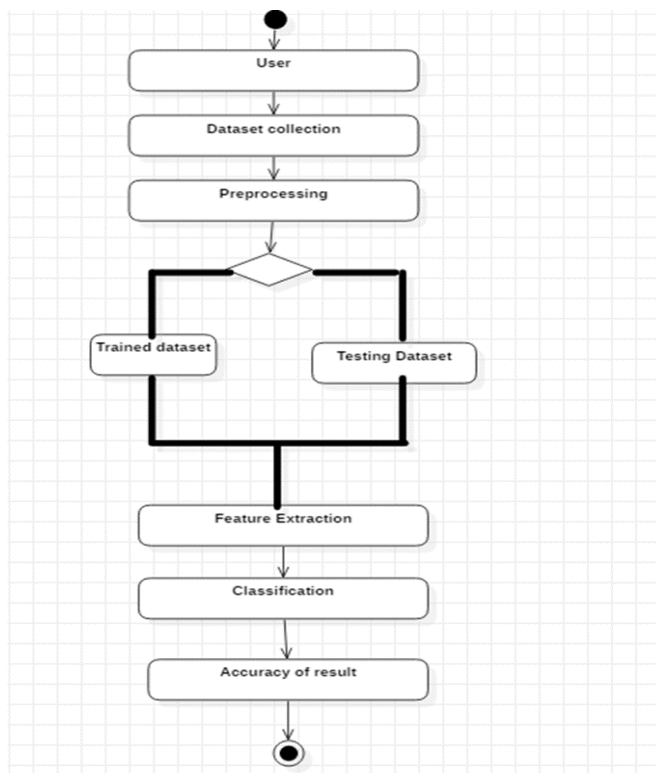
Class Diagram:



Sequence Diagram



Activity Diagram:



6. CONCLUSION AND FUTURE ENHANCEMEN

CONCLUSION :

Interestingly, we have shown that the training of our neural networks can accomplish the highly non-trivial task of determining an optimal estimator to be used as part of the standard hypothesis testing method. This results in a hybrid technique that translates the results of our artificial-intelligence-enabled patient classification algorithm into the language of hypothesis testing, commonly used in biostatistics and medicine, thus establishing a bridge between these two disciplines.. We point out that our research group plans to carry out prospective studies in which the tool presented here is to be applied to future COVID-19 patients as they seek medical attention, including public and private medical facilities with different budget levels located in urban areas with a range of population levels, thus allowing us to evaluate the ability of this new instrument to make useful predictions. We are certain that our work has important implications in the context

REFERENCES

- [1] D. Mazzei, F. Chiarello, and G. Fantoni. (2021). Analyzing social robotics research with natural language processing techniques, *Cognitive Computation*.
- [2] J. Cho and H. Garcia-Molina. (2000). The evolution of the web and implications for an incremental crawler, *Proceedings of the 26th International Conference on Very Large Data Bases, VLDB 00*, 200-209.
- [3] S. Fu, C. C. Wyles, D. R. Osmon, M. L. Carvour, E. Sagheb, T. Ramazanian, W. K. Kremers, D. G. Lewallen, D. J. Berry, S. Sohn, and H. M. Kremers. (2021). Automated detection of periprosthetic joint infections and data elements using natural language processing, *Journal of Arthroplasty*, 36(2), 688-692.
- [4] E. Mower, S. Narayanan, and M. J. Mataric. (2011). A framework for automatic human emotion classification using emotion profiles, *IEEE Transactions on Audio, Speech and Language Processing*, 19(5), 1057-1070.
- [5] A. M. C. H. Attanayake, S. S. N. Perera, and S. Jayasinghe. (2020). Phenomenological modelling of COVID-19 epidemics in Sri Lanka, Italy and Hebei Province of China. *MedRxiv*, 2020(February).

[6] D. Shung, C. Tsay, L. Laine, D. Chang, F. Li, P. Thomas, C. Partridge, M. Simonov, A. Hsiao, J. K. Tay, and A. Taylor. (2020). Early identification of patients with acute gastrointestinal bleeding using natural language processing and decision rules, *Journal of Gastroenterology and Hepatology* (Australia), 1-8

Importing Libraries

```
In [7]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import chart_studio.plotly as py
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, i
init_notebook_mode (connected=True)
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
import xgboost as xgb
from sklearn.metrics import classification_report, confusion_matrix, acc
import plotly.io as pio
import warnings
warnings.filterwarnings('ignore')
```

Loading Dataset

```
In [8]: df=pd.read_csv('Covid-19_Dataset.csv')
```

```
In [9]: df=pd.read_csv('Covid-19_Dataset.csv')
```

```
In [10]: df.shape
```

```
Out[10]: (1085, 20)
```

```
In [6]: df.dtypes
```

```
Out[6]: id                                int64
case_in_country                          float64
reporting date                           object
summary                                  object
location                                 object
country                                 object
gender                                  object
age                                      float64
symptom_onset                           object
If_onset_approximated                    float64
hosp_visit_date                          object
exposure_start                           object
exposure_end                             object
visiting Wuhan                           int64
from Wuhan                               int64
death                                    int64
recovered                                int64
symptom                                  object
```

```
source          object
link            object
dtype: object
```

In [11]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1085 entries, 0 to 1084
Data columns (total 20 columns):
#   Column              Non-Null Count  Dtype
---  -
0   id                   1085 non-null   int64
1   case_in_country     888 non-null    float64
2   reporting date      1084 non-null   object
3   summary             1080 non-null   object
4   location            1085 non-null   object
5   country             1085 non-null   object
6   gender              902 non-null    object
7   age                 843 non-null    float64
8   symptom_onset       563 non-null    object
9   If_onset_approximated 560 non-null    float64
10  hosp_visit_date     507 non-null    object
11  exposure_start      128 non-null    object
12  exposure_end        341 non-null    object
13  visiting Wuhan      1085 non-null   int64
14  from Wuhan          1085 non-null   int64
15  death               1085 non-null   int64
16  recovered           1085 non-null   int64
17  symptom             270 non-null    object
18  source              1085 non-null   object
19  link                1085 non-null   object
dtypes: float64(3), int64(5), object(12)
memory usage: 169.7+ KB
```

Data Cleaning

First we remove unnecessary columns.

In [12]:

```
df.drop(['id','case_in_country','summary','symptom_onset', 'If_onset_app',
'exposure_end', 'symptom', 'source', 'link'],axis=1,inplace=True)
```

In [13]:

```
100*df.isnull().sum()/df.shape[0]
```

Out[13]:

```
reporting date    0.092166
location          0.000000
country           0.000000
gender            16.866359
age               22.304147
visiting Wuhan    0.000000
from Wuhan        0.000000
death             0.000000
recovered         0.000000
dtype: float64
```

In [14]:

```
df['age']= df['age']. fillna(df['age'].mean())
```

In [15]:

```
df_dum=pd.get_dummies(df['gender'].dropna(), drop_first=True)
```

```
In [16]: df_dum['male'].median()
```

```
Out[16]: 1.0
```

So we fill the null values in gender column with 'male'.

```
In [17]: df['gender'] = df['gender'].fillna('male')
```

Fianlly the columns that contain less than 0.5% null values are remained, and we drop the rows of df that contain these null values.

```
In [18]: df.dropna(inplace=True)
```

```
In [19]: df.isnull().sum()
```

```
Out[19]: reporting date    0
location                0
country                 0
gender                  0
age                     0
visiting Wuhan          0
from Wuhan               0
death                   0
recovered                0
dtype: int64
```

Feature Engineering

We rename some of the columns.

```
In [20]: df.columns = df.columns.str.lower().str.replace(' ', '_')
```

Now we change the format of date and add month column to df.

```
In [21]: df['reporting_date'] = pd.to_datetime(df['reporting_date'])
```

```
In [22]: df['year'] = df['reporting_date'].apply(lambda x: x.year)
df['month'] = df['reporting_date'].apply(lambda x: x.month)
```

```
In [23]: df['month'].unique()
```

```
Out[23]: array([2, 1], dtype=int64)
```

As there is only one year, we can remove the year column.

```
In [24]: df.drop(['reporting_date', 'year'], axis=1, inplace=True)
```

```
In [25]: df.head(2)
```

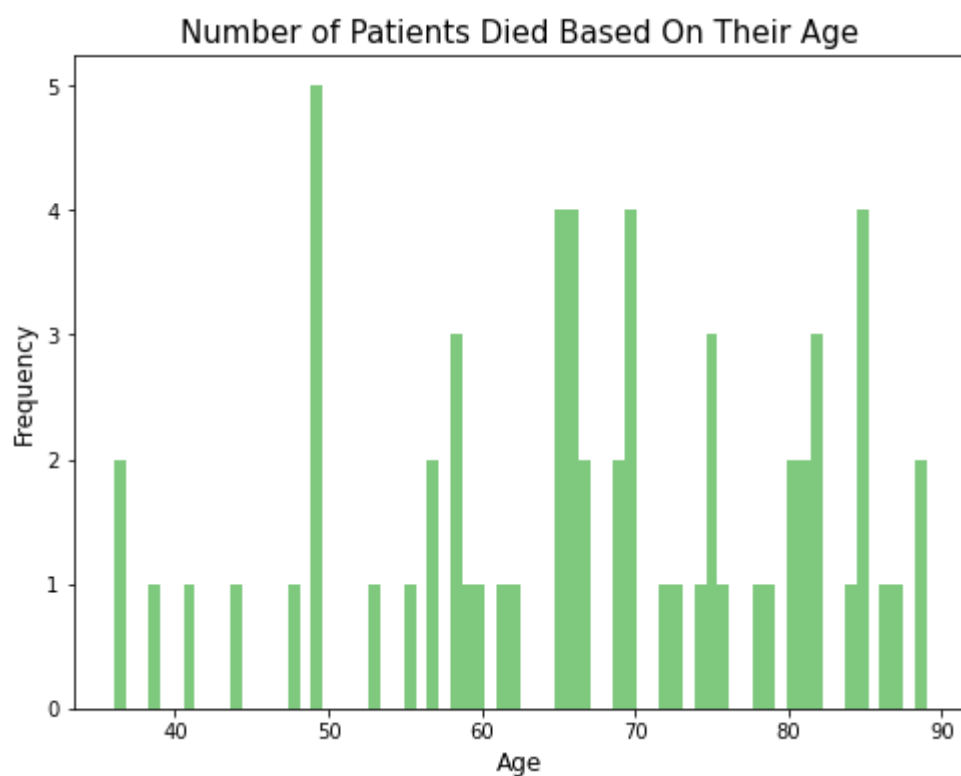
```
Out[25]:
```

	location	country	gender	age	visiting_wuhan	from_wuhan	death	recovered	mont
0	Vinh Phuc	Vietnam	male	0.25	0	0	0	1	
1	Singapore	Singapore	male	0.50	0	0	0	1	

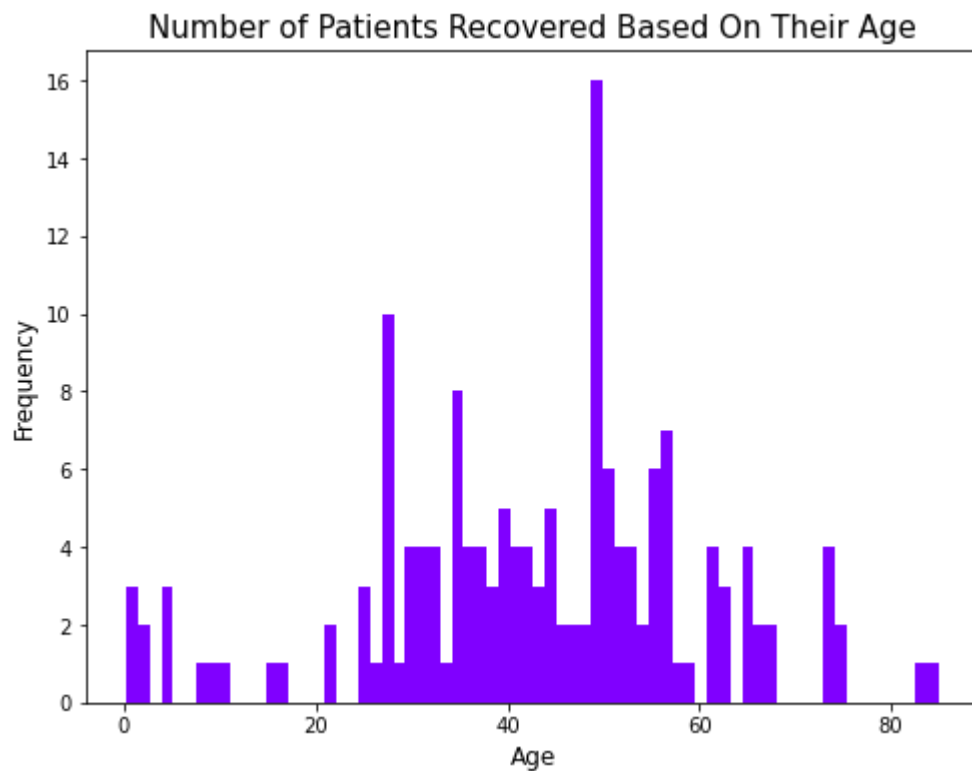
Exploratory Data Analysis

We want to find the number of patients died and recovered based on their age.

```
In [21]: plt.figure(figsize=(8,6))
df[df['death']==1]['age'].plot(kind='hist',bins=70,colormap='Accent')
plt.title('Number of Patients Died Based On Their Age',fontsize=15)
plt.xlabel('Age',fontsize=12)
plt.ylabel('Frequency',fontsize=12)
plt.show()
```



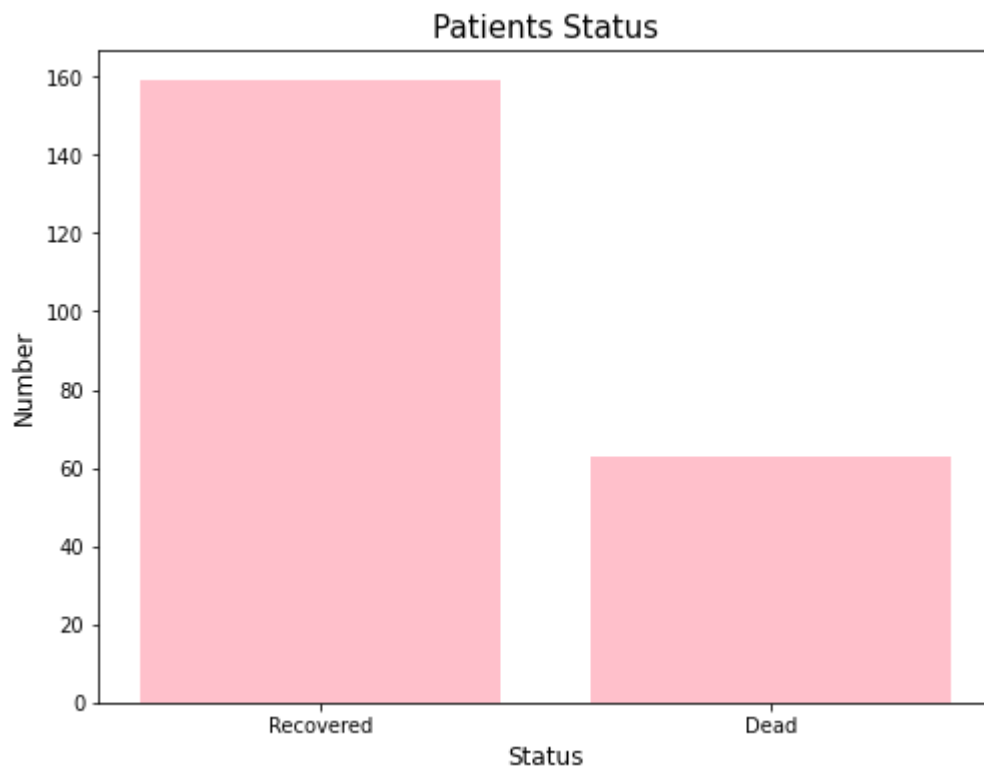
```
In [22]: plt.figure(figsize=(8,6))
df[df['recovered']==1]['age'].plot(kind='hist',bins=70,colormap='rainbow')
plt.title('Number of Patients Recovered Based On Their Age',fontsize=15)
plt.xlabel('Age',fontsize=12)
plt.ylabel('Frequency',fontsize=12)
plt.show()
```



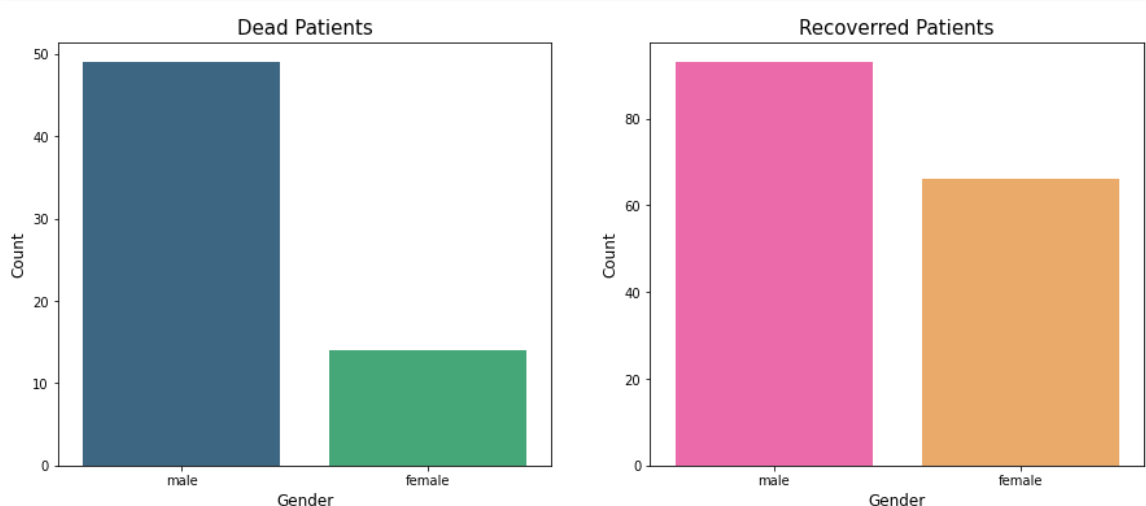
```
In [26]: print('Current count of patients:',df['death'].count())
print('Number of Dead Patients:', df[df['death']==1]['death'].count())
print('Number of Recovered Patients:',df[df['recovered']==1]['death'].count())
print('Number of Patients Receiving Treatment:',df[(df['death']==0)&(df['recovered']==1)].count())
```

```
Current count of patients: 1084
Number of Dead Patients: 63
Number of Recovered Patients: 159
Number of Patients Receiving Treatment: 862
```

```
In [24]: plt.figure(figsize=(8,6))
plt.bar(x=['Recovered','Dead'],height=[159,63], color='pink')
plt.title('Patients Status',fontsize=15)
plt.xlabel('Status', fontsize=12)
plt.ylabel('Number',fontsize=12)
plt.show()
```

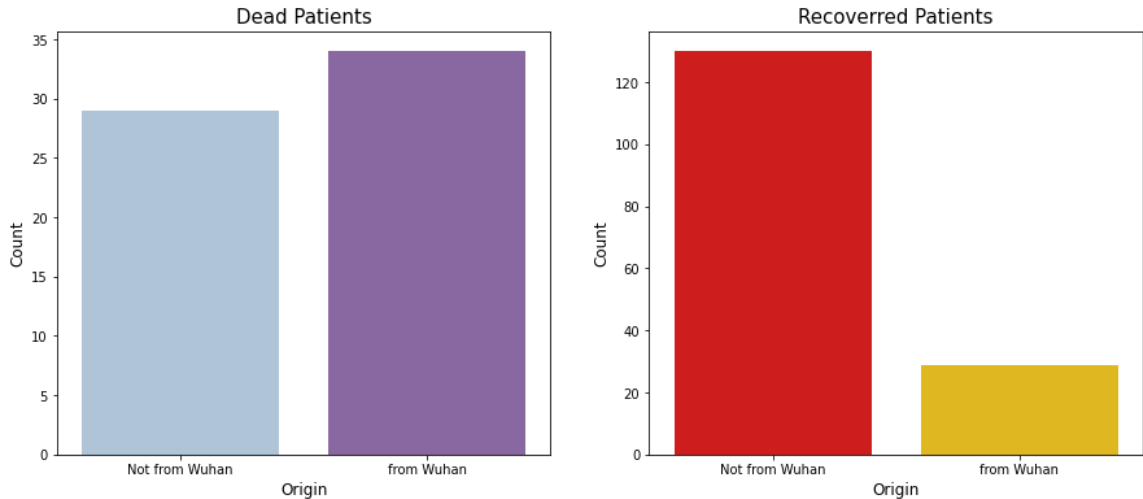


```
In [25]: plt.figure(figsize=(15,6))
plt.subplot(1,2,1)
sns.countplot(x='gender', data=df[df['death']==1], palette='viridis')
plt.xlabel('Gender', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.title('Dead Patients', fontsize=15)
plt.subplot(1,2,2)
sns.countplot(x='gender', data=df[df['recovered']==1], palette='spring')
plt.xlabel('Gender', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.title('Recovered Patients', fontsize=15)
plt.show()
```



```
In [26]: plt.figure(figsize=(15,6))
plt.subplot(1,2,1)
sns.countplot(x='from_wuhan', data=df[df['death']==1], palette='BuPu')
plt.xticks([0,1], ['Not from Wuhan', 'from Wuhan'])
plt.xlabel('Origin', fontsize=12)
plt.ylabel('Count', fontsize=12)
```

```
plt.title('Dead Patients',fontsize=15)
plt.subplot(1,2,2)
sns.countplot(x='from_wuhan', data=df[df['recovered']==1], palette='hot')
plt.xticks([0,1], ['Not from Wuhan','from Wuhan'])
plt.xlabel('Origin', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.title('Recovered Patients',fontsize=15)
plt.show()
```



```
In [27]: country_order=list(df.groupby('country').count()['location'].sort_values
```

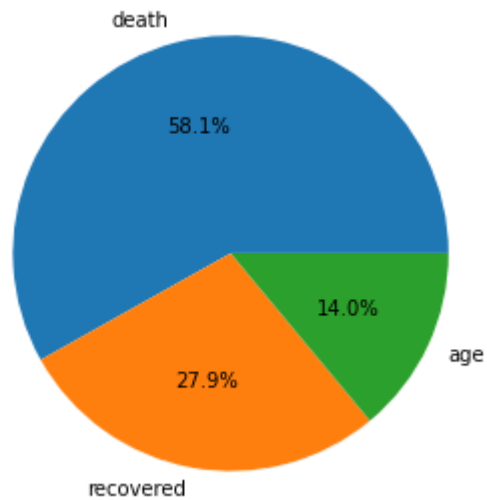
```
In [31]: plt.figure(figsize=(12,6))
sns.countplot(x='country',data=df,color='blue',order=country_order)
plt.xticks(rotation=90)
plt.ylabel('Number of Patients')
plt.xlabel('Country')
plt.title('Number of Covid Patients in Different Countries',fontsize=15)
plt.show()
```

```
-----
--
NameError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_7296\3697975637.py in <module>
      1 plt.figure(figsize=(12,6))
----> 2 sns.countplot(x='country',data=df,color='blue',order=country_order)
      3 plt.xticks(rotation=90)
      4 plt.ylabel('Number of Patients')
      5 plt.xlabel('Country')

NameError: name 'country_order' is not defined
<Figure size 864x432 with 0 Axes>
```

```
In [32]: labels = ["death","recovered","age"]
values = [500,240,120]

f,ax = plt.subplots(figsize=(10, 5))
plt.pie(values, labels = labels,autopct='%1.1f%%')
plt.show()
```



```
In [33]: groupby_df=df.groupby('country').sum()
```

Prediction of Death or Recovery of Patients

Label Encoding

Location

```
In [34]: le1=LabelEncoder()
le1.fit(df['location'])
df['location']=le1.transform(df['location'])
```

Country

```
In [35]: le2=LabelEncoder()
le2.fit(df['country'])
df['country']=le2.transform(df['country'])
```

Gender

```
In [33]: le3=LabelEncoder()
le3.fit(df['gender'])
df['gender']=le3.transform(df['gender'])
```

```
In [34]: df.head()
```

```
Out[34]:
```

	location	country	gender	age	visiting_wuhan	from_wuhan	death	recovered	month
0	141	37	1	0.25	0	0	0	1	2
1	118	26	1	0.50	0	0	0	1	2
2	118	26	1	1.00	0	0	0	1	2
3	42	8	0	2.00	1	0	0	0	1
4	60	22	1	2.00	0	0	0	1	1

Predicting the Recovery

Splitting the Data

```
In [35]: y=df['recovered']
X=df[['location','country','gender','age','visiting_wuhan','from_wuhan',
```

```
In [36]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
```

Random Forest Classifier

```
In [37]: rfc1=RandomForestClassifier(n_estimators=200)
rfc1.fit(X_train,y_train)
predictions_rfc1=rfc1.predict(X_test)
```

```
In [38]: print(confusion_matrix(y_test,predictions_rfc1))
print('\n')
print(classification_report(y_test,predictions_rfc1))
```

```
[[177   8]
 [ 13  19]]
```

	precision	recall	f1-score	support
0	0.93	0.96	0.94	185
1	0.70	0.59	0.64	32
accuracy			0.90	217
macro avg	0.82	0.78	0.79	217
weighted avg	0.90	0.90	0.90	217

```
In [39]: rf_recovery_acc = accuracy_score(y_test,predictions_rfc1)
rf_recovery_acc
```

```
Out[39]: 0.9032258064516129
```

Support Vector Classifier

```
In [40]: svc1=SVC()
svc1.fit(X_train,y_train)
predictions_svc1=svc1.predict(X_test)
```

```
In [41]: print(confusion_matrix(y_test,predictions_svc1))
print('\n')
print(classification_report(y_test,predictions_svc1))
```

```
[[185   0]
 [ 32   0]]
```

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.85	1.00	0.92	185
1	0.00	0.00	0.00	32
accuracy			0.85	217
macro avg	0.43	0.50	0.46	217
weighted avg	0.73	0.85	0.78	217

To improve the results obtained by SVC, we define C and gamma parameters.

```
In [42]: param_grid={'C':[0.1,1,10,100,1000], 'gamma':[1,0.1,0.01,0.001], 'kernel'
```

```
In [43]: gsl=GridSearchCV(SVC(),param_grid, verbose=3)
gsl.fit(X_train,y_train)
predictions_gsl=gsl.predict(X_test)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
[CV 1/5] END .....C=0.1, gamma=1, kernel=rbf;; score=0.851 total time=
0.0s
[CV 2/5] END .....C=0.1, gamma=1, kernel=rbf;; score=0.851 total time=
0.0s
[CV 3/5] END .....C=0.1, gamma=1, kernel=rbf;; score=0.855 total time=
0.0s
[CV 4/5] END .....C=0.1, gamma=1, kernel=rbf;; score=0.855 total time=
0.0s
[CV 5/5] END .....C=0.1, gamma=1, kernel=rbf;; score=0.855 total time=
0.0s
[CV 1/5] END .....C=0.1, gamma=0.1, kernel=rbf;; score=0.851 total time=
0.0s
[CV 2/5] END .....C=0.1, gamma=0.1, kernel=rbf;; score=0.851 total time=
0.0s
[CV 3/5] END .....C=0.1, gamma=0.1, kernel=rbf;; score=0.855 total time=
0.0s
[CV 4/5] END .....C=0.1, gamma=0.1, kernel=rbf;; score=0.855 total time=
0.0s
[CV 5/5] END .....C=0.1, gamma=0.1, kernel=rbf;; score=0.855 total time=
0.0s
[CV 1/5] END .....C=0.1, gamma=0.01, kernel=rbf;; score=0.851 total time=
0.0s
[CV 2/5] END .....C=0.1, gamma=0.01, kernel=rbf;; score=0.851 total time=
0.0s
[CV 3/5] END .....C=0.1, gamma=0.01, kernel=rbf;; score=0.855 total time=
0.0s
[CV 4/5] END .....C=0.1, gamma=0.01, kernel=rbf;; score=0.855 total time=
0.0s
[CV 5/5] END .....C=0.1, gamma=0.01, kernel=rbf;; score=0.855 total time=
0.0s
[CV 1/5] END ....C=0.1, gamma=0.001, kernel=rbf;; score=0.851 total time=
0.0s
[CV 2/5] END ....C=0.1, gamma=0.001, kernel=rbf;; score=0.851 total time=
0.0s
[CV 3/5] END ....C=0.1, gamma=0.001, kernel=rbf;; score=0.855 total time=
0.0s
[CV 4/5] END ....C=0.1, gamma=0.001, kernel=rbf;; score=0.855 total time=
0.0s
[CV 5/5] END ....C=0.1, gamma=0.001, kernel=rbf;; score=0.855 total time=
0.0s
[CV 1/5] END .....C=1, gamma=1, kernel=rbf;; score=0.828 total time=
0.0s
[CV 2/5] END .....C=1, gamma=1, kernel=rbf;; score=0.851 total time=
0.0s
```

[CV 3/5] ENDC=1, gamma=1, kernel=rbf;; score=0.867 total time=0.0s
[CV 4/5] ENDC=1, gamma=1, kernel=rbf;; score=0.855 total time=0.0s
[CV 5/5] ENDC=1, gamma=1, kernel=rbf;; score=0.850 total time=0.0s
[CV 1/5] ENDC=1, gamma=0.1, kernel=rbf;; score=0.845 total time=0.0s
[CV 2/5] ENDC=1, gamma=0.1, kernel=rbf;; score=0.868 total time=0.0s
[CV 3/5] ENDC=1, gamma=0.1, kernel=rbf;; score=0.873 total time=0.0s
[CV 4/5] ENDC=1, gamma=0.1, kernel=rbf;; score=0.879 total time=0.0s
[CV 5/5] ENDC=1, gamma=0.1, kernel=rbf;; score=0.879 total time=0.0s
[CV 1/5] ENDC=1, gamma=0.01, kernel=rbf;; score=0.874 total time=0.0s
[CV 2/5] ENDC=1, gamma=0.01, kernel=rbf;; score=0.874 total time=0.0s
[CV 3/5] ENDC=1, gamma=0.01, kernel=rbf;; score=0.867 total time=0.0s
[CV 4/5] ENDC=1, gamma=0.01, kernel=rbf;; score=0.896 total time=0.0s
[CV 5/5] ENDC=1, gamma=0.01, kernel=rbf;; score=0.861 total time=0.0s
[CV 1/5] ENDC=1, gamma=0.001, kernel=rbf;; score=0.845 total time=0.0s
[CV 2/5] ENDC=1, gamma=0.001, kernel=rbf;; score=0.851 total time=0.0s
[CV 3/5] ENDC=1, gamma=0.001, kernel=rbf;; score=0.855 total time=0.0s
[CV 4/5] ENDC=1, gamma=0.001, kernel=rbf;; score=0.855 total time=0.0s
[CV 5/5] ENDC=1, gamma=0.001, kernel=rbf;; score=0.855 total time=0.0s
[CV 1/5] ENDC=10, gamma=1, kernel=rbf;; score=0.828 total time=0.0s
[CV 2/5] ENDC=10, gamma=1, kernel=rbf;; score=0.851 total time=0.0s
[CV 3/5] ENDC=10, gamma=1, kernel=rbf;; score=0.873 total time=0.0s
[CV 4/5] ENDC=10, gamma=1, kernel=rbf;; score=0.850 total time=0.0s
[CV 5/5] ENDC=10, gamma=1, kernel=rbf;; score=0.855 total time=0.0s
[CV 1/5] ENDC=10, gamma=0.1, kernel=rbf;; score=0.868 total time=0.0s
[CV 2/5] ENDC=10, gamma=0.1, kernel=rbf;; score=0.845 total time=0.0s
[CV 3/5] ENDC=10, gamma=0.1, kernel=rbf;; score=0.850 total time=0.0s
[CV 4/5] ENDC=10, gamma=0.1, kernel=rbf;; score=0.873 total time=0.0s
[CV 5/5] ENDC=10, gamma=0.1, kernel=rbf;; score=0.855 total time=0.0s
[CV 1/5] ENDC=10, gamma=0.01, kernel=rbf;; score=0.856 total time=0.0s
[CV 2/5] ENDC=10, gamma=0.01, kernel=rbf;; score=0.897 total time=0.0s
[CV 3/5] ENDC=10, gamma=0.01, kernel=rbf;; score=0.867 total time=0.0s
[CV 4/5] ENDC=10, gamma=0.01, kernel=rbf;; score=0.884 total time=

0.0s
[CV 5/5] ENDC=10, gamma=0.01, kernel=rbf;; score=0.850 total time=0.0s
[CV 1/5] ENDC=10, gamma=0.001, kernel=rbf;; score=0.851 total time=0.0s
[CV 2/5] ENDC=10, gamma=0.001, kernel=rbf;; score=0.851 total time=0.0s
[CV 3/5] ENDC=10, gamma=0.001, kernel=rbf;; score=0.879 total time=0.0s
[CV 4/5] ENDC=10, gamma=0.001, kernel=rbf;; score=0.873 total time=0.0s
[CV 5/5] ENDC=10, gamma=0.001, kernel=rbf;; score=0.855 total time=0.0s
[CV 1/5] ENDC=100, gamma=1, kernel=rbf;; score=0.828 total time=0.0s
[CV 2/5] ENDC=100, gamma=1, kernel=rbf;; score=0.851 total time=0.0s
[CV 3/5] ENDC=100, gamma=1, kernel=rbf;; score=0.873 total time=0.0s
[CV 4/5] ENDC=100, gamma=1, kernel=rbf;; score=0.850 total time=0.0s
[CV 5/5] ENDC=100, gamma=1, kernel=rbf;; score=0.855 total time=0.0s
[CV 1/5] ENDC=100, gamma=0.1, kernel=rbf;; score=0.856 total time=0.0s
[CV 2/5] ENDC=100, gamma=0.1, kernel=rbf;; score=0.833 total time=0.0s
[CV 3/5] ENDC=100, gamma=0.1, kernel=rbf;; score=0.855 total time=0.0s
[CV 4/5] ENDC=100, gamma=0.1, kernel=rbf;; score=0.867 total time=0.0s
[CV 5/5] ENDC=100, gamma=0.1, kernel=rbf;; score=0.850 total time=0.0s
[CV 1/5] ENDC=100, gamma=0.01, kernel=rbf;; score=0.851 total time=0.0s
[CV 2/5] ENDC=100, gamma=0.01, kernel=rbf;; score=0.862 total time=0.0s
[CV 3/5] ENDC=100, gamma=0.01, kernel=rbf;; score=0.861 total time=0.0s
[CV 4/5] ENDC=100, gamma=0.01, kernel=rbf;; score=0.867 total time=0.0s
[CV 5/5] ENDC=100, gamma=0.01, kernel=rbf;; score=0.827 total time=0.0s
[CV 1/5] ENDC=100, gamma=0.001, kernel=rbf;; score=0.856 total time=0.0s
[CV 2/5] ENDC=100, gamma=0.001, kernel=rbf;; score=0.851 total time=0.0s
[CV 3/5] ENDC=100, gamma=0.001, kernel=rbf;; score=0.879 total time=0.0s
[CV 4/5] ENDC=100, gamma=0.001, kernel=rbf;; score=0.855 total time=0.0s
[CV 5/5] ENDC=100, gamma=0.001, kernel=rbf;; score=0.855 total time=0.0s
[CV 1/5] ENDC=1000, gamma=1, kernel=rbf;; score=0.828 total time=0.0s
[CV 2/5] ENDC=1000, gamma=1, kernel=rbf;; score=0.851 total time=0.0s
[CV 3/5] ENDC=1000, gamma=1, kernel=rbf;; score=0.873 total time=0.0s
[CV 4/5] ENDC=1000, gamma=1, kernel=rbf;; score=0.850 total time=0.0s
[CV 5/5] ENDC=1000, gamma=1, kernel=rbf;; score=0.855 total time=0.0s

```
[CV 1/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.856 total time=0.0s
[CV 2/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.839 total time=0.0s
[CV 3/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.855 total time=0.0s
[CV 4/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.867 total time=0.0s
[CV 5/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.850 total time=0.0s
[CV 1/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.879 total time=0.0s
[CV 2/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.868 total time=0.0s
[CV 3/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.867 total time=0.0s
[CV 4/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.861 total time=0.0s
[CV 5/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.821 total time=0.0s
[CV 1/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.851 total time=0.0s
[CV 2/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.874 total time=0.0s
[CV 3/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.867 total time=0.0s
[CV 4/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.879 total time=0.0s
[CV 5/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.821 total time=0.0s
```

In [44]:

```
print(confusion_matrix(y_test,predictions_gs1))
print('\n')
print(classification_report(y_test,predictions_gs1))
```

```
[[182   3]
 [ 26   6]]
```

	precision	recall	f1-score	support
0	0.88	0.98	0.93	185
1	0.67	0.19	0.29	32
accuracy			0.87	217
macro avg	0.77	0.59	0.61	217
weighted avg	0.84	0.87	0.83	217

In [45]:

```
svr_recovery_acc = accuracy_score(y_test,predictions_gs1)
svr_recovery_acc
```

Out[45]: 0.8663594470046083

XGBoost Classifier

In [46]:

```
xgbcl=xgb.XGBClassifier(n_estimators=200, learning_rate=0.08, gamma=0, s
xgbcl.fit(X_train,y_train)
```

[11:37:12] WARNING: C:/Users/Administrator/workspace/xgboost-win64_releas

e_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.

```
Out[46]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
               colsample_bynode=1, colsample_bytree=1, enable_categorical=
False,
               gamma=0, gpu_id=-1, importance_type=None,
               interaction_constraints='', learning_rate=0.08, max_delta_s
tep=0,
               max_depth=8, min_child_weight=1, missing=nan,
               monotone_constraints='()', n_estimators=200, n_jobs=4,
               num_parallel_tree=1, predictor='auto', random_state=0,
               reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=0.
5,
               tree_method='exact', validate_parameters=1, verbosity=None)
```

```
In [47]: predictions_xgbcl=xgbcl.predict(X_test)
```

```
In [48]: print(confusion_matrix(y_test,predictions_xgbcl))
print('\n')
print(classification_report(y_test,predictions_xgbcl))
```

```
[[180   5]
 [ 11  21]]
```

	precision	recall	f1-score	support
0	0.94	0.97	0.96	185
1	0.81	0.66	0.72	32
accuracy			0.93	217
macro avg	0.88	0.81	0.84	217
weighted avg	0.92	0.93	0.92	217

```
In [49]: xg_recovery_acc = accuracy_score(y_test,predictions_xgbcl)
xg_recovery_acc
```

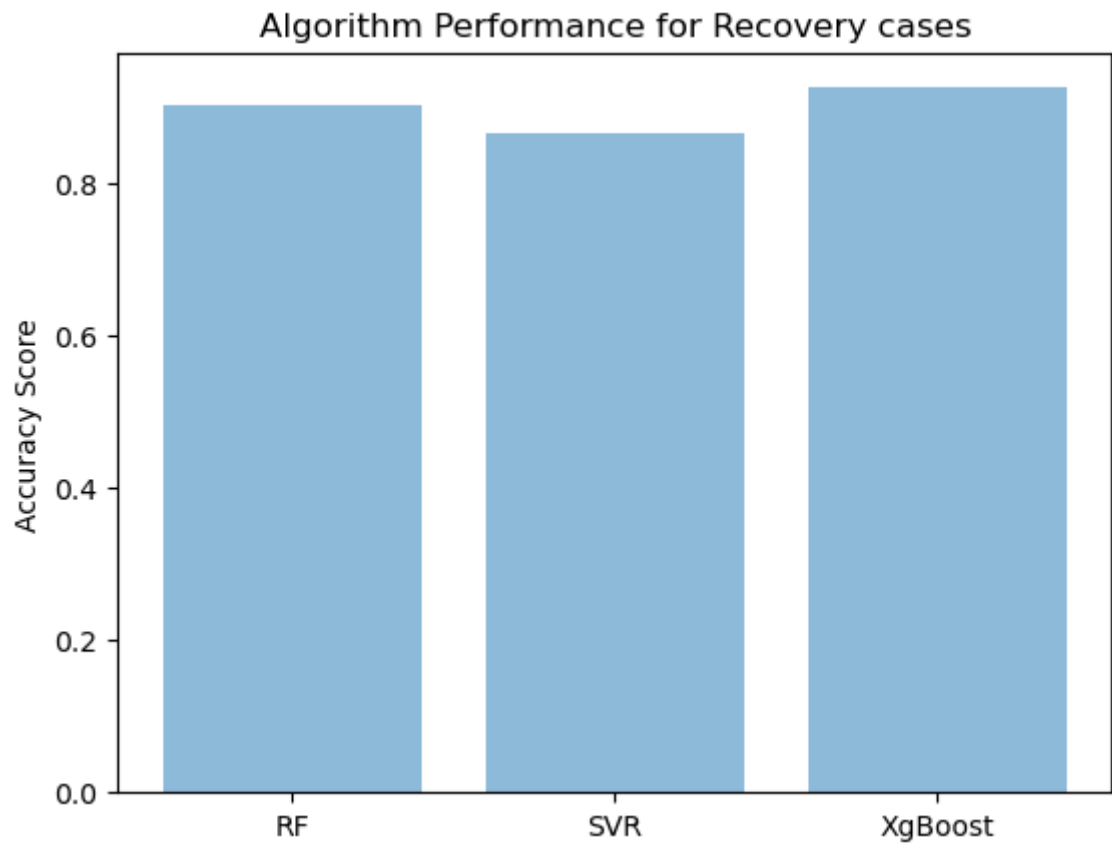
```
Out[49]: 0.9262672811059908
```

```
In [50]: import matplotlib.pyplot as plt; plt.rcParams()
import numpy as np
import matplotlib.pyplot as plt

objects = ('RF','SVR','XgBoost')
y_pos = np.arange(len(objects))
performance = [rf_recovery_acc,svr_recovery_acc,xg_recovery_acc]

plt.bar(y_pos, performance, align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('Accuracy Score')
plt.title('Algorithm Performance for Recovery cases')

plt.show()
```



Predicting the Death

Splitting the Data

```
In [51]: y=df['death']
         x=df[['location','country','gender','age','visiting_wuhan','from_wuhan',
```

```
         x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
```

Random Forest Classifier

```
In [53]: rfc2=RandomForestClassifier(n_estimators=200)
         rfc2.fit(X_train,y_train)
         predictions_rfc2=rfc2.predict(X_test)
```

```
In [54]: print(confusion_matrix(y_test,predictions_rfc2))
         print('\n')
         print(classification_report(y_test,predictions_rfc2))
```

```
[[202  1]
 [  9  5]]
```

	precision	recall	f1-score	support
0	0.96	1.00	0.98	203
1	0.83	0.36	0.50	14
accuracy			0.95	217

macro avg	0.90	0.68	0.74	217
weighted avg	0.95	0.95	0.95	217

```
In [55]: rf_death_acc = accuracy_score(y_test,predictions_rfc2)
rf_death_acc
```

```
Out[55]: 0.9539170506912442
```

Support Vector Classifier

```
In [56]: svc2=SVC()
svc2.fit(X_train,y_train)
predictions_svc2=svc2.predict(X_test)
```

```
In [57]: print(confusion_matrix(y_test,predictions_svc2))
print('\n')
print(classification_report(y_test,predictions_svc2))
```

```
[[203  0]
 [ 14  0]]
```

	precision	recall	f1-score	support
0	0.94	1.00	0.97	203
1	0.00	0.00	0.00	14
accuracy			0.94	217
macro avg	0.47	0.50	0.48	217
weighted avg	0.88	0.94	0.90	217

To improve the results obtained by SVC, we define C and gamma parameters.

```
In [58]: param_grid={'C':[0.1,1,10,100,1000], 'gamma':[1,0.1,0.01,0.001], 'kernel
```

```
In [59]: gs2=GridSearchCV(SVC(),param_grid, verbose=3)
gs2.fit(X_train,y_train)
predictions_gs2=gs2.predict(X_test)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
[CV 1/5] END .....C=0.1, gamma=1, kernel=rbf;; score=0.943 total time=
0.0s
[CV 2/5] END .....C=0.1, gamma=1, kernel=rbf;; score=0.943 total time=
0.0s
[CV 3/5] END .....C=0.1, gamma=1, kernel=rbf;; score=0.948 total time=
0.0s
[CV 4/5] END .....C=0.1, gamma=1, kernel=rbf;; score=0.942 total time=
0.0s
[CV 5/5] END .....C=0.1, gamma=1, kernel=rbf;; score=0.942 total time=
0.0s
[CV 1/5] END .....C=0.1, gamma=0.1, kernel=rbf;; score=0.943 total time=
0.0s
[CV 2/5] END .....C=0.1, gamma=0.1, kernel=rbf;; score=0.943 total time=
0.0s
[CV 3/5] END .....C=0.1, gamma=0.1, kernel=rbf;; score=0.948 total time=
```

0.0s
[CV 4/5] ENDC=0.1, gamma=0.1, kernel=rbf;; score=0.942 total time=0.0s
[CV 5/5] ENDC=0.1, gamma=0.1, kernel=rbf;; score=0.942 total time=0.0s
[CV 1/5] ENDC=0.1, gamma=0.01, kernel=rbf;; score=0.943 total time=0.0s
[CV 2/5] ENDC=0.1, gamma=0.01, kernel=rbf;; score=0.943 total time=0.0s
[CV 3/5] ENDC=0.1, gamma=0.01, kernel=rbf;; score=0.948 total time=0.0s
[CV 4/5] ENDC=0.1, gamma=0.01, kernel=rbf;; score=0.942 total time=0.0s
[CV 5/5] ENDC=0.1, gamma=0.01, kernel=rbf;; score=0.942 total time=0.0s
[CV 1/5] ENDC=0.1, gamma=0.001, kernel=rbf;; score=0.943 total time=0.0s
[CV 2/5] ENDC=0.1, gamma=0.001, kernel=rbf;; score=0.943 total time=0.0s
[CV 3/5] ENDC=0.1, gamma=0.001, kernel=rbf;; score=0.948 total time=0.0s
[CV 4/5] ENDC=0.1, gamma=0.001, kernel=rbf;; score=0.942 total time=0.0s
[CV 5/5] ENDC=0.1, gamma=0.001, kernel=rbf;; score=0.942 total time=0.0s
[CV 1/5] ENDC=1, gamma=1, kernel=rbf;; score=0.948 total time=0.0s
[CV 2/5] ENDC=1, gamma=1, kernel=rbf;; score=0.954 total time=0.0s
[CV 3/5] ENDC=1, gamma=1, kernel=rbf;; score=0.960 total time=0.0s
[CV 4/5] ENDC=1, gamma=1, kernel=rbf;; score=0.960 total time=0.0s
[CV 5/5] ENDC=1, gamma=1, kernel=rbf;; score=0.942 total time=0.0s
[CV 1/5] ENDC=1, gamma=0.1, kernel=rbf;; score=0.966 total time=0.0s
[CV 2/5] ENDC=1, gamma=0.1, kernel=rbf;; score=0.971 total time=0.0s
[CV 3/5] ENDC=1, gamma=0.1, kernel=rbf;; score=0.977 total time=0.0s
[CV 4/5] ENDC=1, gamma=0.1, kernel=rbf;; score=0.971 total time=0.0s
[CV 5/5] ENDC=1, gamma=0.1, kernel=rbf;; score=0.965 total time=0.0s
[CV 1/5] ENDC=1, gamma=0.01, kernel=rbf;; score=0.977 total time=0.0s
[CV 2/5] ENDC=1, gamma=0.01, kernel=rbf;; score=0.971 total time=0.0s
[CV 3/5] ENDC=1, gamma=0.01, kernel=rbf;; score=0.971 total time=0.0s
[CV 4/5] ENDC=1, gamma=0.01, kernel=rbf;; score=0.965 total time=0.0s
[CV 5/5] ENDC=1, gamma=0.01, kernel=rbf;; score=0.960 total time=0.0s
[CV 1/5] ENDC=1, gamma=0.001, kernel=rbf;; score=0.971 total time=0.0s
[CV 2/5] ENDC=1, gamma=0.001, kernel=rbf;; score=0.960 total time=0.0s
[CV 3/5] ENDC=1, gamma=0.001, kernel=rbf;; score=0.965 total time=0.0s
[CV 4/5] ENDC=1, gamma=0.001, kernel=rbf;; score=0.971 total time=0.0s

[CV 5/5] ENDC=1, gamma=0.001, kernel=rbf;; score=0.954 total time=0.0s
[CV 1/5] ENDC=10, gamma=1, kernel=rbf;; score=0.954 total time=0.0s
[CV 2/5] ENDC=10, gamma=1, kernel=rbf;; score=0.954 total time=0.0s
[CV 3/5] ENDC=10, gamma=1, kernel=rbf;; score=0.965 total time=0.0s
[CV 4/5] ENDC=10, gamma=1, kernel=rbf;; score=0.960 total time=0.0s
[CV 5/5] ENDC=10, gamma=1, kernel=rbf;; score=0.948 total time=0.0s
[CV 1/5] ENDC=10, gamma=0.1, kernel=rbf;; score=0.977 total time=0.0s
[CV 2/5] ENDC=10, gamma=0.1, kernel=rbf;; score=0.971 total time=0.0s
[CV 3/5] ENDC=10, gamma=0.1, kernel=rbf;; score=0.971 total time=0.0s
[CV 4/5] ENDC=10, gamma=0.1, kernel=rbf;; score=0.965 total time=0.0s
[CV 5/5] ENDC=10, gamma=0.1, kernel=rbf;; score=0.942 total time=0.0s
[CV 1/5] ENDC=10, gamma=0.01, kernel=rbf;; score=0.983 total time=0.0s
[CV 2/5] ENDC=10, gamma=0.01, kernel=rbf;; score=0.971 total time=0.0s
[CV 3/5] ENDC=10, gamma=0.01, kernel=rbf;; score=0.971 total time=0.0s
[CV 4/5] ENDC=10, gamma=0.01, kernel=rbf;; score=0.977 total time=0.0s
[CV 5/5] ENDC=10, gamma=0.01, kernel=rbf;; score=0.965 total time=0.0s
[CV 1/5] ENDC=10, gamma=0.001, kernel=rbf;; score=0.977 total time=0.0s
[CV 2/5] ENDC=10, gamma=0.001, kernel=rbf;; score=0.966 total time=0.0s
[CV 3/5] ENDC=10, gamma=0.001, kernel=rbf;; score=0.965 total time=0.0s
[CV 4/5] ENDC=10, gamma=0.001, kernel=rbf;; score=0.965 total time=0.0s
[CV 5/5] ENDC=10, gamma=0.001, kernel=rbf;; score=0.954 total time=0.0s
[CV 1/5] ENDC=100, gamma=1, kernel=rbf;; score=0.954 total time=0.0s
[CV 2/5] ENDC=100, gamma=1, kernel=rbf;; score=0.954 total time=0.0s
[CV 3/5] ENDC=100, gamma=1, kernel=rbf;; score=0.965 total time=0.0s
[CV 4/5] ENDC=100, gamma=1, kernel=rbf;; score=0.960 total time=0.0s
[CV 5/5] ENDC=100, gamma=1, kernel=rbf;; score=0.948 total time=0.0s
[CV 1/5] ENDC=100, gamma=0.1, kernel=rbf;; score=0.983 total time=0.0s
[CV 2/5] ENDC=100, gamma=0.1, kernel=rbf;; score=0.971 total time=0.0s
[CV 3/5] ENDC=100, gamma=0.1, kernel=rbf;; score=0.971 total time=0.0s
[CV 4/5] ENDC=100, gamma=0.1, kernel=rbf;; score=0.965 total time=0.0s
[CV 5/5] ENDC=100, gamma=0.1, kernel=rbf;; score=0.948 total time=0.0s
[CV 1/5] ENDC=100, gamma=0.01, kernel=rbf;; score=0.977 total time=

```

0.0s
[CV 2/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.971 total time=
0.0s
[CV 3/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.954 total time=
0.0s
[CV 4/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.971 total time=
0.0s
[CV 5/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.948 total time=
0.0s
[CV 1/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.983 total time=
0.0s
[CV 2/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.977 total time=
0.0s
[CV 3/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.971 total time=
0.0s
[CV 4/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.983 total time=
0.0s
[CV 5/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.960 total time=
0.0s
[CV 1/5] END .....C=1000, gamma=1, kernel=rbf;, score=0.954 total time=
0.0s
[CV 2/5] END .....C=1000, gamma=1, kernel=rbf;, score=0.954 total time=
0.0s
[CV 3/5] END .....C=1000, gamma=1, kernel=rbf;, score=0.965 total time=
0.0s
[CV 4/5] END .....C=1000, gamma=1, kernel=rbf;, score=0.960 total time=
0.0s
[CV 5/5] END .....C=1000, gamma=1, kernel=rbf;, score=0.948 total time=
0.0s
[CV 1/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.983 total time=
0.0s
[CV 2/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.971 total time=
0.0s
[CV 3/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.971 total time=
0.0s
[CV 4/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.965 total time=
0.0s
[CV 5/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.948 total time=
0.0s
[CV 1/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.977 total time=
0.0s
[CV 2/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.971 total time=
0.0s
[CV 3/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.954 total time=
0.0s
[CV 4/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.977 total time=
0.0s
[CV 5/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.948 total time=
0.0s
[CV 1/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.977 total time=
0.0s
[CV 2/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.977 total time=
0.0s
[CV 3/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.977 total time=
0.0s
[CV 4/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.983 total time=
0.0s
[CV 5/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.960 total time=
0.0s

```

```

In [60]: print(confusion_matrix(y_test,predictions_gs2))
          print('\n')

```

```
print(classification_report(y_test,predictions_gs2))
```

```
[[203  0]
 [  8  6]]
```

	precision	recall	f1-score	support
0	0.96	1.00	0.98	203
1	1.00	0.43	0.60	14
accuracy			0.96	217
macro avg	0.98	0.71	0.79	217
weighted avg	0.96	0.96	0.96	217

```
In [61]: svr_death_acc = accuracy_score(y_test,predictions_gs2)
svr_death_acc
```

```
Out[61]: 0.9631336405529954
```

XGBoost Classifier

```
In [62]: xgbc2=xgb.XGBClassifier(n_estimators=200, learning_rate=0.08, gamma=0, s
xgbc2.fit(X_train,y_train)
```

[11:37:17] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.

```
Out[62]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
colsample_bynode=1, colsample_bytree=1, enable_categorical=
False,
gamma=0, gpu_id=-1, importance_type=None,
interaction_constraints='', learning_rate=0.08, max_delta_s
tep=0,
max_depth=8, min_child_weight=1, missing=nan,
monotone_constraints='()', n_estimators=200, n_jobs=4,
num_parallel_tree=1, predictor='auto', random_state=0,
reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=0.
5,
tree_method='exact', validate_parameters=1, verbosity=None)
```

```
In [63]: predictions_xgbc2=xgbc2.predict(X_test)
```

```
In [64]: print(confusion_matrix(y_test,predictions_xgbc2))
print('\n')
print(classification_report(y_test,predictions_xgbc2))
```

```
[[203  0]
 [  9  5]]
```

	precision	recall	f1-score	support
0	0.96	1.00	0.98	203
1	1.00	0.36	0.53	14

accuracy			0.96	217
macro avg	0.98	0.68	0.75	217
weighted avg	0.96	0.96	0.95	217

```
In [65]: xg_death_acc = accuracy_score(y_test,predictions_xgbc2)
xg_death_acc
```

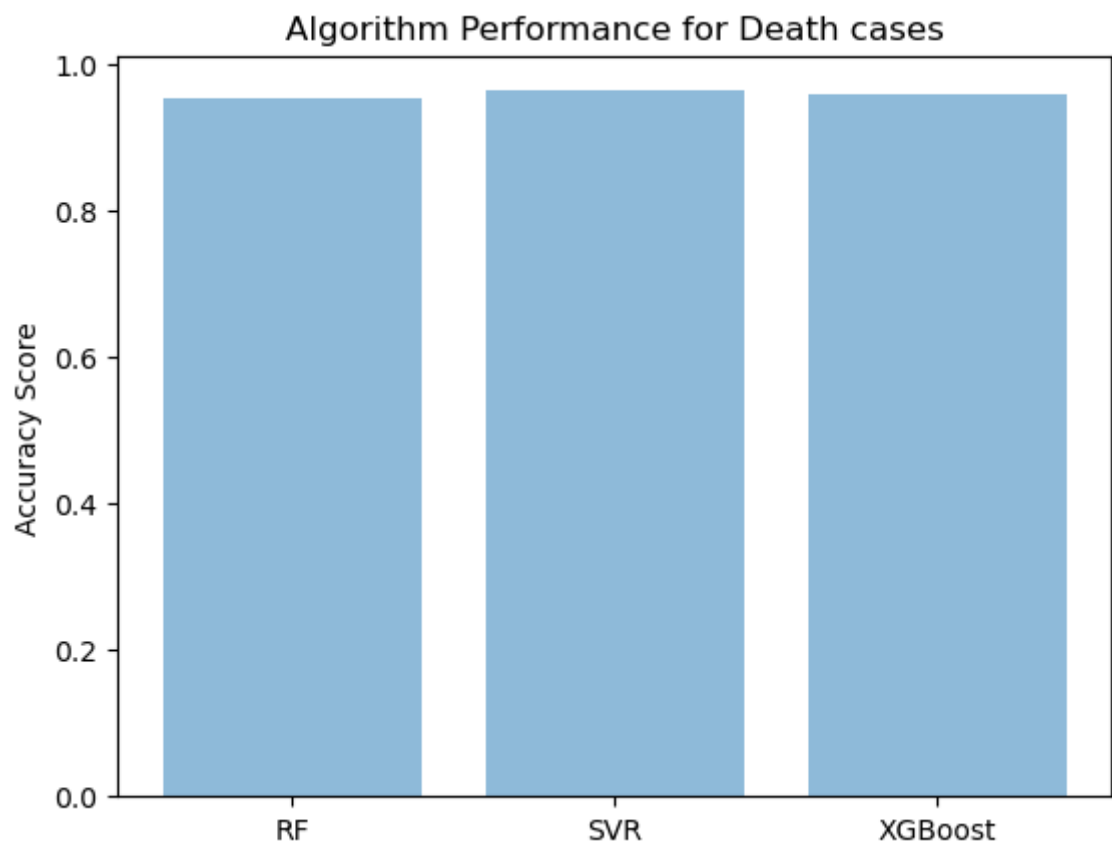
```
Out[65]: 0.9585253456221198
```

```
In [66]: import matplotlib.pyplot as plt; plt.rcParams()
import numpy as np
import matplotlib.pyplot as plt

objects = ('RF','SVR','XGBoost')
y_pos = np.arange(len(objects))
performance = [rf_death_acc,svr_death_acc,xg_death_acc]

plt.bar(y_pos, performance, align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('Accuracy Score')
plt.title('Algorithm Performance for Death cases')

plt.show()
```



```
In [ ]:
```

```
In [ ]:
```