ABSTRACT

Data mining is a process used by companies to turn raw data into useful information. By using software to look for patterns in large batches of data, businesses can learn more about their customers to develop more effective marketing strategies, increase sales and decrease costs. Data mining depends on effective data collection, warehousing, and computer processing, artificial intelligence (e.g., machine learning) and business intelligence. Data mining involves exploring and analysing large blocks of information to glean meaningful patterns and trends. It can be used in a variety of ways, such as database marketing, credit risk management, fraud detection, spam Email filtering, or even to discern the sentiment or opinion of users.

The data mining process breaks down into five steps. First, organizations collect data and load it into their data warehouses. Next, they store and manage the data, either on in-house servers or the cloud. Business analysts, management teams and information technology professionals access the data and determine how they want to organize it. Then, application software sorts the data based on the user's results, and finally, the end-user presents the data in an easy-to-share format, such as a graph or table

In our project we have performed data mining on a dataset of COVID -19 and predicted some important results related to it. COVID – 19 is a disease caused due to corona virus. Coronaviruses primarily infect the upper respiratory and gastrointestinal tract of mammals and birds.

In December 2019, a pneumonia outbreak was reported in Wuhan, China. On 31 December 2019, the outbreak was traced to a novel strain of coronavirus, which was given the interim name 2019-nCoV by the World Health Organization (WHO), later renamed SARS-CoV-2 by the International Committee on Taxonomy of Viruses. Some researchers have suggested the Wuhan Seafood Wholesale Market may not be the original source of viral transmission to humans. As of 14 April 2020, there have been at least 119,686 confirmed deaths and more than 1,920,918 confirmed cases in the coronavirus pneumonia pandemic. The Wuhan strain has been identified as a new strain of Beta coronavirus from group 2B with approximately 70% genetic similarity to the SARS-CoV. The virus has a 96% similarity to a bat coronavirus, so it is widely suspected to originate from bats as well. The pandemic has resulted in travel restrictions and nationwide lockdowns in several countries.

This project is also submitted at this **Kaggle kernel**.

1.1 Description

We've made a project "**Covid-19 Analysis"** in which we try to analyze the causes/effects of the recent pandemic that is Coronavirus (COVID-19). Dataset has been used that consists of various attributes. Using these attributes, we try to find correlations so that proper analysis of the situation can be made. We have used Python programming language and various in-built libraries of python for the purpose of Data Mining and Analysis.

Our system will consist of 5 stages: Data Preprocessing (Data cleaning, transformation and reduction), Encoding, Feature Selection, Visualization and finally Prediction. We have used ***logistic* regression** machine learning algorithm on the dataset and predicted the accuracy score by training the model. It would predict the chance of Death or Recovery of a patient depending upon the attributes (input variables).

We built this project with the purpose of analyzing the cause and effect of COVID-19. It would help to state proper facts and relations and removing any fake relations prevailing in the society.

The dataset contains several parameters which were recorder based on the patients identified at different point of time. The parameters included are :

- Reporting date – date of patient identified
- Location – location of patient
- Country – country in which patient is identified
- Gender – gender of patient (Male-1, Female-0)
- Age – age of patient
- Visiting Wuhan – have the patient visited Wuhan in recent time
- From Wuhan – is the patient from Wuhan
- Death – does the patient died because of Covid-19 (0 or 1)
- Recovered – does the patient recovered from Covid-19 (0 or 1)

Dataset - The dataset used in this notebook (Covid-19_dataset.csv) is same as the COVID19_line_list_data.csv dataset taken from https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset, with the only difference that in our dataset death and recovered columns contains values in the form of 0 or 1 and not in form of dates as in the later dataset.

How it works?

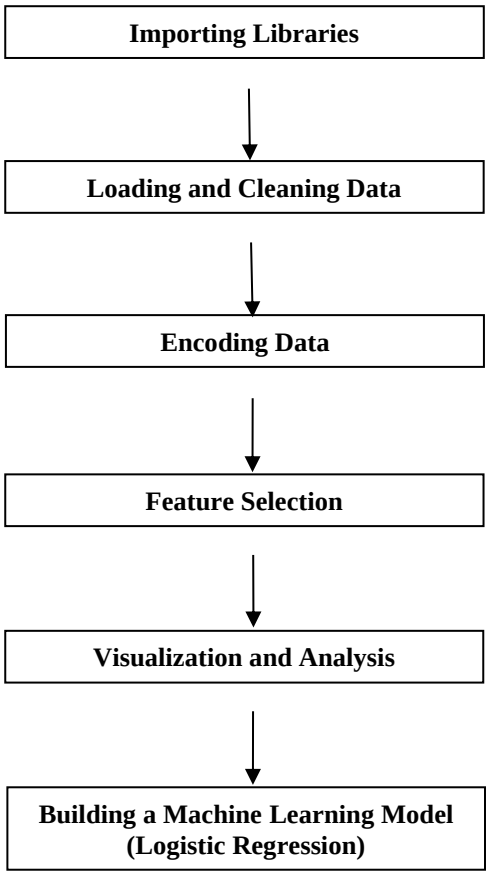


Figure 8. Workflow of COVID-19 Analysis

1.2 Importing Libraries

First load the required libraries.

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import plotly.express as px
        from sklearn.preprocessing import LabelEncoder
        from sklearn.feature_selection import SelectKBest
        from sklearn.feature_selection import chi2
        from sklearn.ensemble import ExtraTreesClassifier
        from sklearn.linear_model import LogisticRegression
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import mean_squared_error
        from math import sqrt
        from sklearn import metrics
        from sklearn.metrics import classification_report
```

1.3 Loading Data

Now, it's moment to load data, here I have used that location where dataset is stored in my system. The name of dataset is "Covid-19_dataset.csv". For displaying in more representative way it is converted to data frame using Pandas library of Python after processing data.

```
In [2]: data = pd.DataFrame(pd.read_csv("Covid-19_dataset.csv"))
        data.head(2)
```

Out[2]:

	id	case_in_country	reporting date	summary	location	country	gender	age	symptom_onset	If_onset_approximated	hosp_visit_date	exposure_start	ex
0	765	15.0	02-10-20	new confirmed COVID-19 patient in Vietnam: 3 m...	Vinh Phuc	Vietnam	NaN	0.25	NaN	NaN	NaN	NaN	
1	477	27.0	02-05-20	new confirmed COVID-19 patient in Singapore: m...	Singapore	Singapore	male	0.50	NaN	NaN	NaN	1/23/2020	

1.4 Cleaning Data

Data Cleansing is a form of data management. Over time, persons and businesses mount up a lot of personal information! Ultimately, information becomes out-of-date. For example, over 10 years you may transform your address, or your name, and then change your address again! Data cleansing is a method in which you go through all of the data within a database and either eradicate or modernize information that is incomplete, mistaken, improperly formatted, irrelevant, or duplicated. Data cleansing usually involves cleaning up data compiled in one area. For example, data from a single spreadsheet.

Though data cleansing does and can engross deleting information, it is focused more on updating, correcting, and consolidating data to ensure your system is as effective as possible.

```
In [3]: data.columns = data.columns.str.strip().str.lower().str.replace(' ', '_')
        data.head(2)
```

Out[3]:

	id	case_in_country	reporting_date	summary	location	country	gender	age	symptom_onset	if_onset_approximated	hosp_visit_date	exposure_star
0	765	15.0	02-10-20	new confirmed COVID-19 patient in Vietnam: 3 m...	Vinh Phuc	Vietnam	NaN	0.25	NaN	NaN	NaN	NaN
1	477	27.0	02-05-20	new confirmed COVID-19 patient in Singapore: m...	Singapore	Singapore	male	0.50	NaN	NaN	NaN	1/23/202(

```
In [5]: data.drop(data.columns[[0,1,3,8,9,10,11,12,17,18,19]], axis = 1, inplace = True)
        data['reporting_date'] = pd.to_datetime(data.reporting_date)

        data.head()
```

Out[5]:

	reporting_date	location	country	gender	age	visiting_wuhan	from_wuhan	death	recovered
0	2020-02-10	Vinh Phuc	Vietnam	NaN	0.25	0	0	0	1
1	2020-02-05	Singapore	Singapore	male	0.50	0	0	0	1
2	2020-02-17	Singapore	Singapore	male	1.00	0	0	0	1
3	2020-01-25	Hechi, Guangxi	China	female	2.00	1	0	0	0
4	2020-01-25	Johor	Malaysia	male	2.00	0	0	0	1

```
In [6]: data.shape
Out[6]: (1085, 9)

In [7]: data.columns
Out[7]: Index(['reporting_date', 'location', 'country', 'gender', 'age',
               'visiting_wuhan', 'from_wuhan', 'death', 'recovered'],
              dtype='object')
```

isnull () Detect missing values - Return a Boolean same-sized object indicating if the values are NA. NA values, such as None or numpy.NaN, gets mapped to True values. Everything else gets mapped to False values. Characters such as empty strings '' or numpy.inf are not considered NA values (unless you set pandas.options.mode.use_inf_as_na = True).

```
In [10]: print('Number of Null values in Columns')
         data.isnull().sum()

         Number of Null values in Columns

Out[10]: reporting_date      1
         location            0
         country             0
         gender            183
         age               242
         visiting_wuhan      0
         from_wuhan          0
         death               0
         recovered           0
         dtype: int64
```

Deleting Rows with Null Values in Specific columns

```
In [11]: refined_data = data.dropna(subset=['gender', 'age', 'from_wuhan'])
```

```
In [12]: print('Number of Null values in Columns')
         refined_data.isnull().sum()
```

```
Number of Null values in Columns

Out[12]: reporting_date    0
         location          0
         country           0
         gender            0
         age               0
         visiting_wuhan    0
         from_wuhan        0
         death             0
         recovered         0
         dtype: int64
```

.**head() and .tail()** - Head and tail functions are capable of 5 rows per time. But you can change this situation. So you can enter the desired value in the parameter section. The first function, ie head (), returns the initial values. The second function returns the last values.

describe() - Describe function includes analysis of all our numerical data. For this, count, mean, std, min,% 25,% 50,% 75, max values are given. The reason this section is important is that you can estimate the probability that the values found here are deviant data.

info() - Then, the data has what informations. We are learning the information for all data The info function shows the data types and numerical values of the features in our data set. In short, this information about our data set. :)

In [13]:
```
refined_data.head(5)
```

Out[13]:

	reporting_date	location	country	gender	age	visiting_wuhan	from_wuhan	death	recovered
1	2020-02-05	Singapore	Singapore	male	0.5	0	0	0	1
2	2020-02-17	Singapore	Singapore	male	1.0	0	0	0	1
3	2020-01-25	Hechi, Guangxi	China	female	2.0	1	0	0	0
4	2020-01-25	Johor	Malaysia	male	2.0	0	0	0	1
5	2020-02-10	Singapore	Singapore	female	2.0	1	0	0	1

In [69]:
```
refined_data.shape
```

Out[69]:
```
(825, 9)
```

In [70]:
```
refined_data.columns
```

Out[70]:
```
Index(['reporting_date', 'location', 'country', 'gender', 'age',
       'visiting_wuhan', 'from_wuhan', 'death', 'recovered'],
      dtype='object')
```

In [15]:
```
refined_data.describe()
```

Out[15]:

	age	visiting_wuhan	from_wuhan	death	recovered
count	825.000000	825.000000	825.000000	825.000000	825.000000
mean	49.755758	0.179394	0.186667	0.070303	0.173333
std	17.991875	0.383915	0.389880	0.255812	0.378765
min	0.500000	0.000000	0.000000	0.000000	0.000000
25%	35.000000	0.000000	0.000000	0.000000	0.000000
50%	51.000000	0.000000	0.000000	0.000000	0.000000
75%	64.000000	0.000000	0.000000	0.000000	0.000000
max	96.000000	1.000000	1.000000	1.000000	1.000000

In [16]:
```
refined_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 825 entries, 1 to 1079
Data columns (total 9 columns):
reporting_date    825 non-null datetime64[ns]
location          825 non-null object
country           825 non-null object
gender            825 non-null object
age               825 non-null float64
visiting_wuhan    825 non-null int64
from_wuhan        825 non-null int64
death             825 non-null int64
recovered         825 non-null int64
dtypes: datetime64[ns](1), float64(1), int64(4), object(3)
memory usage: 64.5+ KB
```

1.5 Encoding Data

In machine learning, we usually deal with datasets which contains multiple labels in one or more than one columns. These labels can be in the form of words or numbers. To make the data understandable or in human readable form, the training data is often labeled in words. Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

SPECIFIC LOCATION IN A COUNTRY

In [17]:
```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
refined_data[refined_data.columns[1]] = labelencoder.fit_transform(refined_data[refined_data.columns[1]])
```

COUNTRY

In [18]:
```
labelencoder = LabelEncoder()
refined_data[refined_data.columns[2]] = labelencoder.fit_transform(refined_data[refined_data.columns[2]])
```

GENDER

In [19]:
```
labelencoder = LabelEncoder()
refined_data[refined_data.columns[3]] = labelencoder.fit_transform(refined_data[refined_data.columns[3]])
```

1.6 Feature Selection

- **Univariate Selection -** Statistical tests can be used to select those features that have the strongest relationship with the output variable.

  The scikit-learn library provides the SelectKBest class that can be used with a suite of different statistical tests to select a specific number of features.

  The example below uses the chi-squared (chi²) statistical test for non-negative features to select 5 of the best features from the Dataset.

In [28]:
```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

X = refined_data.iloc[:,1:7]  #independent columns
y = refined_data.iloc[:,7]    #target column i.e Death
```

In [29]:
```
#apply SelectKBest class to extract top 5 best features
bestfeatures = SelectKBest(score_func=chi2, k=5)
fit = bestfeatures.fit(X,y)
```

In [30]:
```
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
```

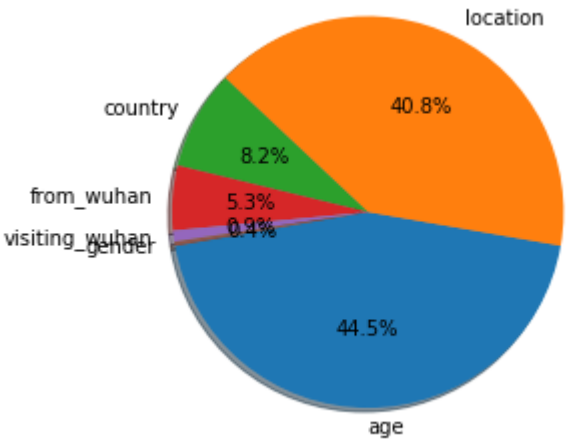
In [31]:
```
#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']  #naming the dataframe columns
featureScores
```

Out[31]:

	Specs	Score
0	location	407.528544
1	country	81.670406
2	gender	3.567871
3	age	444.595027
4	visiting_wuhan	9.143794
5	from_wuhan	53.350788

In [32]:
```
print(featureScores.nlargest(6,'Score'))
```

```
              Specs       Score
3               age  444.595027
0          location  407.528544
1           country   81.670406
5         from_wuhan   53.350788
4     visiting_wuhan    9.143794
2            gender    3.567871
```



Therefore, we can conclude that Deaths are mostly effected by the 'Location' and 'Age' of the Patient as those features have the highest Feature Score.

- **Feature Importance -** Feature importance gives you a score for each feature of our data, the higher the score more important or relevant is the feature towards your output variable.

  Feature importance is an inbuilt class that comes with Tree Based Classifiers, we will be using Extra Tree Classifier for extracting the top 5 features for the dataset.

```
In [34]: from sklearn.ensemble import ExtraTreesClassifier
         import matplotlib.pyplot as plt
         model = ExtraTreesClassifier()
         model.fit(X,y)
```

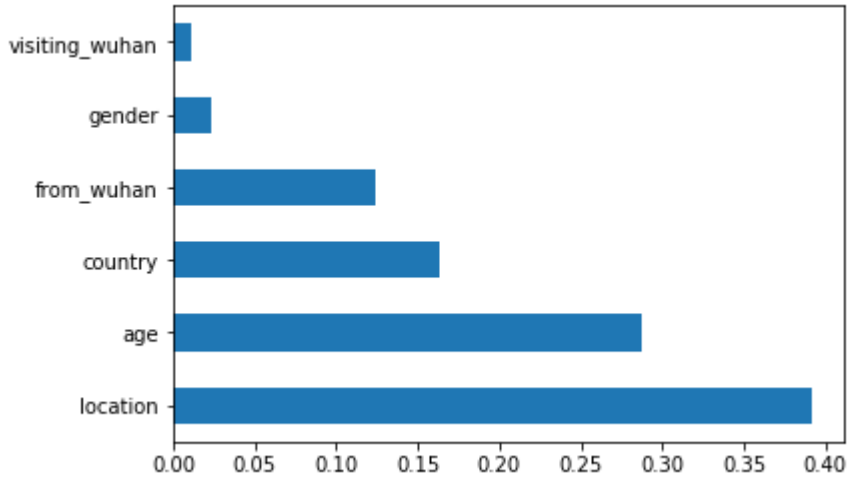
```
C:\Users\71guk\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning:

The default value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
```

```
Out[34]: ExtraTreesClassifier(bootstrap=False, class_weight=None, criterion='gini',
                     max_depth=None, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
                     oob_score=False, random_state=None, verbose=0, warm_start=False)
```

```
In [35]: print(model.feature_importances_) #use inbuilt class feature_importances of tree based classifiers
```

```
[0.39153005 0.16301164 0.02326351 0.28698329 0.01068951 0.12452199]
```

```
In [36]: #plot graph of feature importances for better visualization
         feat_importances = pd.Series(model.feature_importances_, index=X.columns)
         feat_importances.nlargest(10).plot(kind='barh')
         plt.show()
```

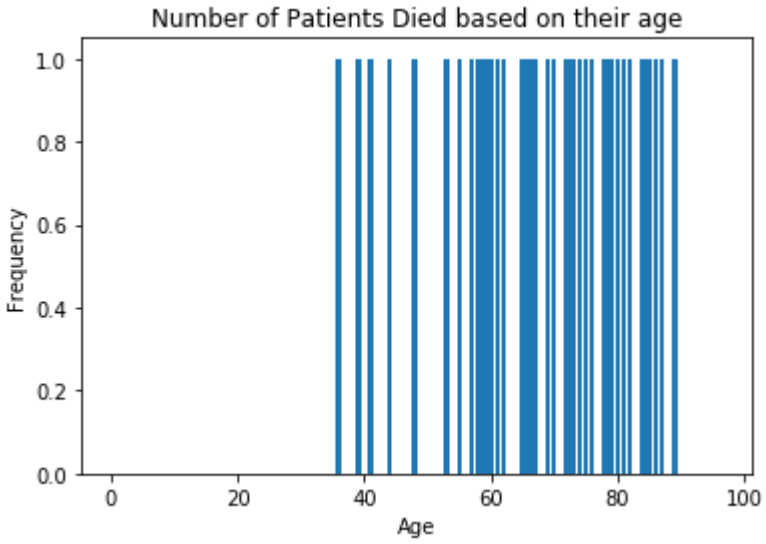


We can conclude that 'Gender' of the patient or Whether they have 'visited Wuhan' has very little impact on their chance of Death.

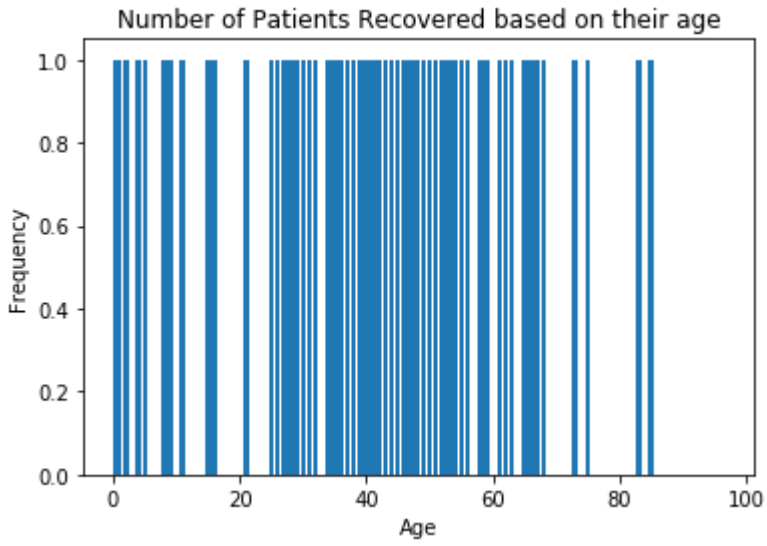
1.7 Visualization and Analysis

We will perform analysis on the training data. The relationship between the features found in the training data is observed. In this way, comments about the properties can be made.

- We can conclude that Patients of age more than 55 have high chance of death because of their weaker immune system.

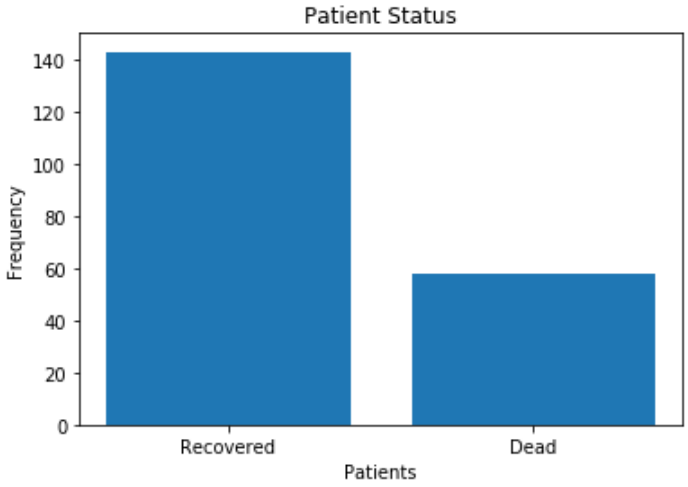


- Patients of age below 55 have high chance of recovery.

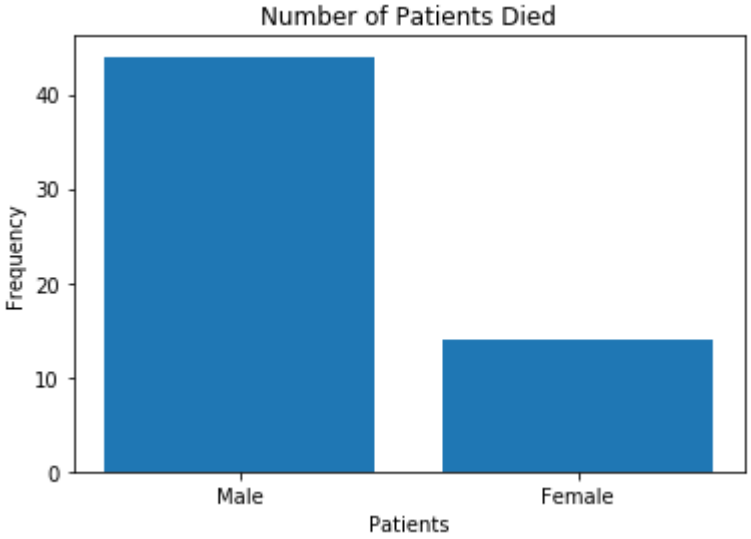


- From the following graphs we can conclude that the number of patients recovered is more than the number of patients died.

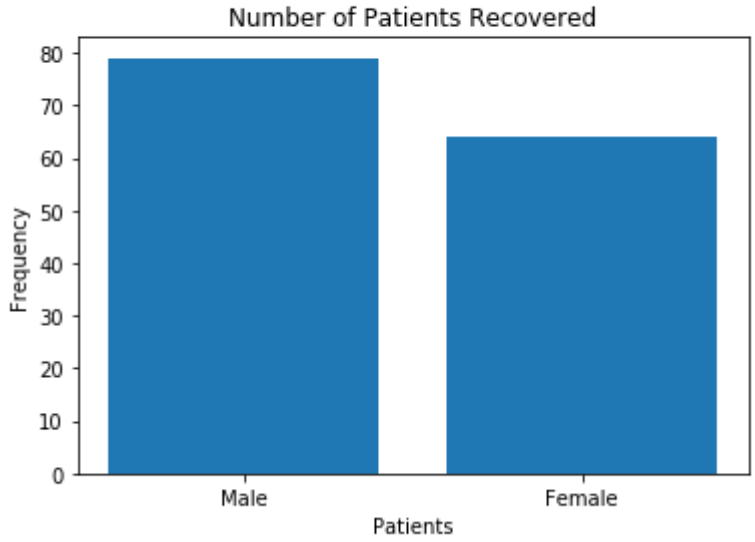
```
Current count of patients: 825
Recovered: 143
Dead: 58
Number of Patients receiving treatment: 624
```



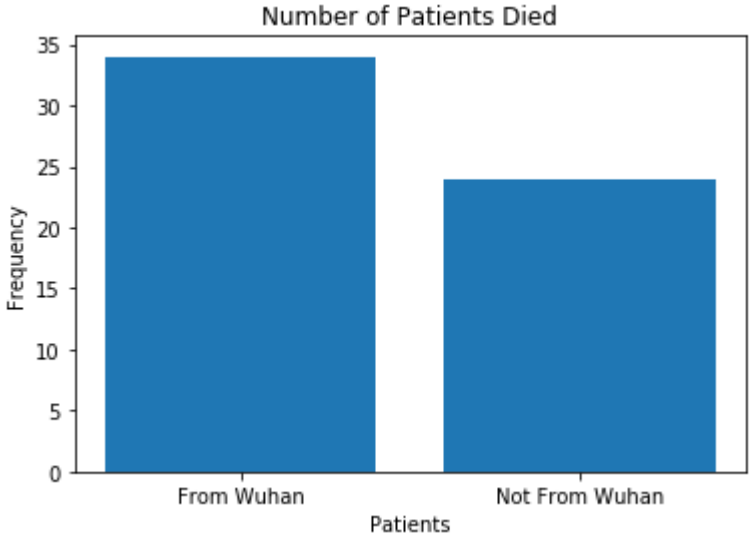
- As more Males die as well as recover with comparison to Females, therefore this data is redundant and we cannot predict patient's health based on their gender.

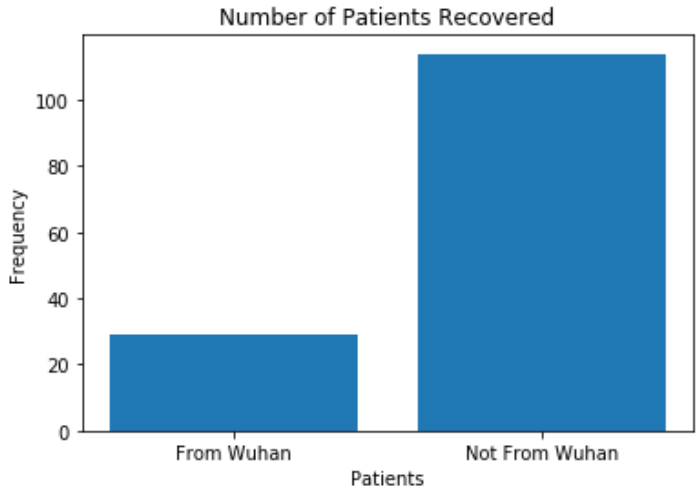


```
Male: 79
Female: 64
```

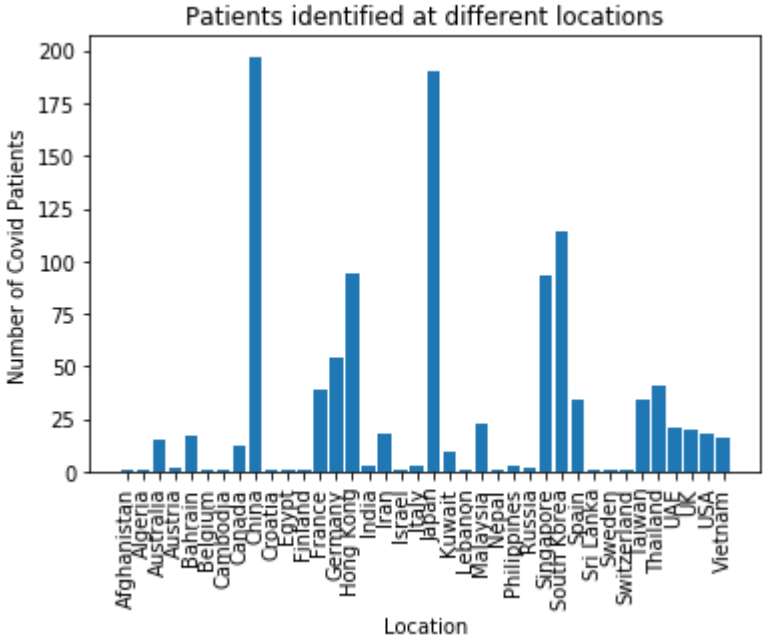


- Patients from Wuhan have high chance of death then recovery as compared to patients not from Wuhan.

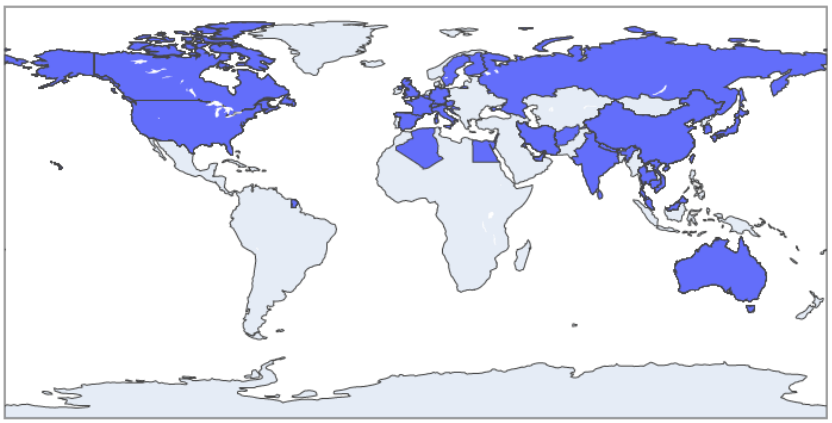




- Certain locations have more patients as compared to others. Countries close to China and those countries where more tourists are found have higher count of patients.



PATIENTS IDENTIFIED AT DIFFERENT LOCATIONS



1.8 Building the Model

Modeling in machine learning is an interactive phase where a data scientist constantly train and test machine learning models to find out the most excellent one for the given task.

Different machine learning models exist and the selection of which one to use generally depends on the problem at hand. No machine learning model works best for all kind of

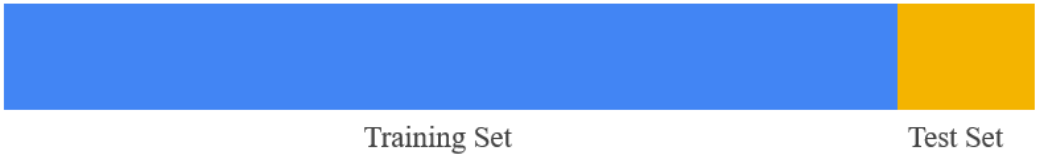
problems. So, your job in this stage is to test multiple models and fine tune parameters to compress out every bit of accuracy.

Here, we have used **Logistic Regression -** as the output variables (death/recovery) is of the form 0 or 1.

Splitting Data:- In this step dataset is split into two parts, that is, training set and testing set.

- **Training Set:-** A subset to train model.
- **Testing Set:-** A subset to test the trained model.

You could picturize slicing the single data set as follows:



- **Predicting DEATH of a Patient**

```
In [54]: X = refined_data[refined_data.columns[1:7]] #(location, country, gender, age, visiting wuhan, from wuhan)
         y = refined_data[refined_data.columns[[7]]] #death

In [55]: from sklearn.linear_model import LogisticRegression
         from sklearn.model_selection import train_test_split
         X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)

         reg=LogisticRegression()
         reg.fit(X_train,y_train)

         C:\Users\71guk\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning:

         Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.

         C:\Users\71guk\Anaconda3\lib\site-packages\sklearn\utils\validation.py:761: DataConversionWarning:

         A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
         el().

Out[55]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='warn',
                   n_jobs=None, penalty='l2', random_state=None, solver='warn',
                   tol=0.0001, verbose=0, warm_start=False)
```

Accuracy of our model predicting death is – 94.7%

```
In [57]: reg.score(X_train,y_train)
Out[57]: 0.946969696969697
```

```
In [59]: #CONFUSION MATRIX
         cm = metrics.confusion_matrix(y_test, pdt)
         print(cm)

         [[154   0]
          [  9   2]]
```

```
In [60]: from sklearn.metrics import mean_squared_error
         from math import sqrt

         rms = sqrt(mean_squared_error(y_test,pdt))
         rms

Out[60]: 0.23354968324845687
```

```
In [61]: from sklearn.metrics import classification_report
         print(classification_report(y_test, pdt))
```

	precision	recall	f1-score	support
0	0.94	1.00	0.97	154
1	1.00	0.18	0.31	11
micro avg	0.95	0.95	0.95	165
macro avg	0.97	0.59	0.64	165
weighted avg	0.95	0.95	0.93	165

- **Predicting RECOVERY of a Patient**

```
In [62]: X = refined_data[refined_data.columns[1:7]] #(location, country, gender, age, visiting wuhan, from wuhan)
         y = refined_data[refined_data.columns[[8]]] #recovered
```

```
In [63]: X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)

         reg=LogisticRegression()
         reg.fit(X_train,y_train)

         C:\Users\71guk\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning:

         Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.

         C:\Users\71guk\Anaconda3\lib\site-packages\sklearn\utils\validation.py:761: DataConversionWarning:

         A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
         el().

Out[63]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='warn',
                   n_jobs=None, penalty='l2', random_state=None, solver='warn',
                   tol=0.0001, verbose=0, warm_start=False)
```

Accuracy of our model predicting recovery is – 81.121%

```
In [64]: reg.score(X_train,y_train)

Out[64]: 0.8121212121212121
```

In [66]:
```
#CONFUSION MATRIX
cm = metrics.confusion_matrix(y_test, pdt)
print(cm)
```

```
[[138   5]
 [ 19   3]]
```

In [67]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt

rms = sqrt(mean_squared_error(y_test,pdt))
rms
```

Out[67]: 0.3813850356982369

In [68]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, pdt))
```

```
              precision    recall  f1-score   support

           0       0.88      0.97      0.92       143
           1       0.38      0.14      0.20        22

   micro avg       0.85      0.85      0.85       165
   macro avg       0.63      0.55      0.56       165
weighted avg       0.81      0.85      0.82       165
```

CONCLUSION

2019 Novel Coronavirus (2019-nCoV) is a virus (more specifically, a coronavirus) identified as the cause of an outbreak of respiratory illness first detected in Wuhan, China. Early on, many of the patients in the outbreak in Wuhan, China reportedly had some link to a large seafood and animal market, suggesting animal-to-person spread. However, a growing number of patients reportedly have not had exposure to animal markets, indicating person-to-person spread is occurring. At this time, it's unclear how easily or sustainably this virus is spreading between people.

So daily level information on the affected people can give some interesting insights when it is made available to the broader data science community.

From our analysis we have concluded that based on the dataset we have taken that the patients health who is suffering from COVID-19 depends on certain conditions and is independent from others. For example, a patient of age 75 has a high chance of death than recovery. We know exceptions are always, our analysis is based on the dataset only. Certain factors which are more important in determining and predicting a patient's heath are –

- **Patient's Age**
  Patients of age more than 55 have high chance of death because of their weaker immune system and Patients of age below 55 have high chance of recovery.

- **Patient's Location**
  Countries close to China and those countries where more tourists are found have higher count of patients.

- **Is the Patient from Wuhan as it was the primary hotspot for COVID-19**
  Patents who are from Wuhan have higher chance of death than recovery.

And certain factors such as **Gender** is of no use based on the data as no conclusive evidence is there that a patient's recovery or death depends on their gender.

Recovery of a patient depends upon their immune system and this will be the case till a vaccine is developed for COVID-19.

REFERENCES

- www.kaggle.com
- https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset
- https://www.mygov.in/covid-19
- https://www.worldometers.info/coronavirus/?
- https://www.covid19india.org/
- www.datacamp.com
- www.towardsdatascence.com
- www.machinelearningmastery.com