Prediction of COVID-19 patients' survival

Syeda Narmeen

190699987

Group size: 3

Table of individual contribution by each member of my group, based on my subjective opinion:

Student	Effort
Marvi Memon	33.3%
Narmeen Syeda	33.3%
Amina Quraishi	33.3%

Table of Contents

1 Introduction and Background

No historic event in the lives of the world's majority has ever compared to the one being witnessed currently, the COVID-19 Global Pandemic. Originating in Wuhan, China in December 2019, it swiftly spread across the world [4], leaving behind approximately 2,595 deaths within two months and was classified as a Public Health Emergency of International Concern by the World Health Organisation [4]. With repercussions ranging far and wide, few have been exempted from its physical, mental, spiritual and financial effects.

Although the research currently being conducted to fully comprehend transmission, susceptibility of infection, and probability of survival is occurring at a rapid pace, the acceleration of COVID-19's transmission rates and it's devastating effects produces a formidable challenge to researchers. Meanwhile, great underlying insecurity combined with the lack of understanding of the virus is causing major concern. Current figures as of April 30, 2020 estimate the global death toll at 228,000 [1]. Although these numbers of deaths pale in comparison to the estimated 50 to 100 million deaths from the Spanish Flu in 1918 [2], the ambiguity within its domain is causing severe impacts.

Governments have taken numerous mitigating interventions which has slowed the transmission of COVID-19, however, there is more to understand about this mysterious disease.

Until scientists become familiar with this new and previously unknown virus, a dearth of information and scientific research will continue to persist; thus, COVID-19 topics remain an interesting and undeveloped area of research at this moment, in particular, facilitating the topic of how severity of infection and other factors may lead to morbidity.

2 Project Aims

Our research aims to predict survival of a patient who has been infected by COVID-19. Through an exploration of possible morbidity risk factors, we would like to deduce if there are variables which increase the infected patient's probability of survival and if we can predict the death of a patient with COVID-19.

We will be performing the initial exploration and visualisation of the dataset using a Jupyter notebook for an easy interface and using Python3 language to implement instructions onto the dataset. Following this step, we aim to implement a supervised classification learning model to perform the predictive analysis. Using the XGBoost and K-Nearest-Neighbours classifiers, we will use different success matrices to assess the 'success' of our model including the accuracy, precision, recall and F1 score. Along with this, we will use the confusion matrix to visualise the false positives and false negatives and the receiver operating characteristic (ROC) curve.

One caveat to mention here is that due to the unfamiliarity with this novel virus, the COVID-19 data is constantly being updated, where multiple contributors may propose varying research on a daily basis. The ongoing changing data can affect the accuracy and outcome each time the model is run of this research paper's results. The data we were able to retrieve is one where we found the most relevant information for our research. It includes numerous different variables including demographics. Nonetheless, it is important to acknowledge limitations within the obtained dataset. There are a high number of null values, which limits comparisons as well as a significant amount of unknown values.

3 Literature Review

Significant research has been conducted for this new, emergent virus named COVID-19 in such a short time-frame, however, due to its existence being no longer than a few months, it is integral to compare and analyse the information we have for this virus, to a similar pre-existing virus. SARS (Sudden Acute Respiratory Syndrome) or MERS (Middle Eastern Respiratory Syndrome), are undoubtedly the closest to this new contagion. Genetic, demographic, clinical and temporal similarities between these comparative viral epidemics and COVID-19 support discussion of SARS in order to attempt an understanding of the new virus. SARS provides clues as to the influence of these morbidity risk factors (MRFs) on the survival rate of COVID-19 patients. One may infer that COVID-19 will follow a similar trajectory in several areas discussed above.

The overwhelmingly crucial MRF that was securely established in both SARS and the limited COVID-19 research studies was old age, with its various lower threshold limits, usually greater than age sixty years old [6][8][9][10][12][13]. The next most common type of MRF was discovered to be co-morbidities [9][10][11][12][13][14][15]. In one study, the majority of patients that had died from COVID-19 exhibited "significant health conditions including hypertension, diabetes, heart and/or kidney function issues that may have made them more susceptible [9]". The effects of these co-morbidities on survival are analysed in this study. Thirdly, the subsequent most common type of MRF is sex; although the male sex was initially a statistically significant MRF in several studies, it was later found through multiple regression analysis and other analysis that sex does not play a significant role in determining survival [6][9][10][11][12]. This study will analyze this MRF as well.

Apart from these three variables (age, co-morbidity, and sex), research on either disease suggests various other MRFs: genetic mutations of the virus, producing different strains [5][9], immunological differences in particular populations such as those exhibiting mutated host gene receptors such as ACE2 for the virus to attach to [5, 15], connection to Wuhan [10], incubation periods of the onset to the death of an infected patient, currently estimated at 6 to 41 days with a median of 14 days [11][13][16], a number of medical test results such as Vitamin D levels indicating the status of the patient's immune system, lymphopenia, and D-dimer levels [4][10][11][13]. Among these MRFs, the connection to Wuhan's seafood market was found to indicate an association to a majority of COVID-19 cases; however, recent cases reveal a lack of direct connections to Wuhan [9]. In this study, we investigate this MRF in the available dataset which may prove or disprove its feature importance.

Epidemic prevention has been focused on blocking the infection resource, but has now shifted to the "prompt identification and prediction of critically ill cases prone to death by the clinical and laboratory features [9]." It is in this manner that interventions may be successful to avoid fatalities.

4 Data Retrieval

1. Data source and description

The dataset for the analysis was downloaded from the git repository for a project named "Epidemiological Data from the nCoV-2019 Outbreak: Early Descriptions from Publicly Available Data". This data set has also been used to create COVID-19 -19 health maps. Multiple contributors update this dataset daily [20].
The obtained dataset has raw data within a file named *latest.csv*. It contains COVID-19 patients' details such as their gender, age, country , travel history, chronic diseases, hospital history, connection to Wuhan, and the outcome. The outcome of patient health status varies from being stable or critical to death or discharged.

This dataset is updated on a daily basis. At the time of writing this paper, the dataset was updated 2 days ago and had the shape of (476126, 33). The columns in the dataset are the following:

```
Index(['ID', 'age', 'sex', 'city', 'province', 'country', 'latitude',
       'longitude', 'geo_resolution', 'date_onset_symptoms',
       'date_admission_hospital', 'date_confirmation', 'symptoms',
       'lives_in_Wuhan', 'travel_history_dates', 'travel_history_location',
       'reported_market_exposure', 'additional_information',
       'chronic_disease_binary', 'chronic_disease', 'source',
       'sequence_available', 'outcome', 'date_death_or_discharge',
       'notes_for_discussion', 'location', 'admin3', 'admin2', 'admin1',
       'country_new', 'admin_id', 'data_moderator_initials',
       'travel_history_binary'],
      dtype='object')
```

5 Data Representation

The coding for this paper was carried out in Python 3 as it has numerous useful and powerful libraries and packages required to carry out different data analysis techniques. The libraries used are:

Numpy: A general-purpose array processing package. It provides a high-performance multi-dimensional array object, and high-level maths functions for operation on these arrays.

Pandas: A high-level data manipulation tool that is built on top of the Numpy package. We have used pandas to read data from the *.csv* file and store in the data frame. The data frame allows us to store data as a 2D matrix and provides various operations for data manipulation and cleaning.

Matplotlib: A library that allows creating static, animated, and interactive visualization such as scatter plot, bar charts, maps, etc. We used it to create visuals for our data to better understand and explore our data, as well as to provide visualization of our machine-learning models.

Seaborn: Another library for visuals based on matplotlib, but this additionally provides a high-level interface for drawing attractive and informative statistical graphics. It is closely integrated with pandas and data structures. We used it for data exploration.

Scikit-learn: A library that provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. The library is built upon Scipy that needs to install before using scikit-learn.

6 Data Cleaning and Feature Engineering

6.1 Removing Duplicates

Within the dataset, it was mandatory to remove any duplicated rows that were present. We did so and this resulted in no dropping of columns indicating that there were no replicates of rows.

6.2 Dropping Columns

The data set consisted of 33 columns. Hence the first thing we noticed was the fact that there were quite a few repetitive columns and columns that, in our opinion, would not be needed. This included 'longitude', 'latitude', 'geo_resolution', 'source', 'notes_for_discussion', 'admin_id', 'admin1', 'admin2', 'admin3, 'data_moderator_initials', sequence_available', 'city' and 'province'. Majority of these were string columns, and did not mean much to us. Although 'city' and 'province' could've been used, there were multiple variations, and hence we decided to just keep the 'country' which we could use much more easily.

6.3 Modifying Columns

Multiple columns were modified in the process of exploration of the data. 'Sex' was made into a binary column with females being 1 and males being 0. Along with this, we had a column labeled 'lives_in_wuhan', which we chose to change to 'con_to_wuhan' meaning, connection to Wuhan. This also reflected back onto our research that mentioned that even though the Huanan seafood market in Wuhan has been associated with the majority of cases, many of the recent cases do not have a direct connection [4]. We wanted to create this column to test whether or not this was one of our top features. This also felt necessary as it covered people who worked in wuhan as well and we could hence convert this into a binary format for the machine learning algorithm. We also converted the 'travel_history_binary' and 'chronic_disease_binary' into 1's and 0's instead of having true and false. We finally set the index to the 'ID' for each patient.

We modified the age column which we felt was essential. This column was by far one of the hardest we had to tidy up. The column incorporated strings and consisted of ranges as well. We all agreed on taking the mean of the ranges and then bucketing the ages into 5 different categories: unknown/blank, child, adolescent, adult and senior. All values below 0, over 100, or blank were categorised as unknown, 0-9 child, 10-19 an adolescent, 20-59 an adult, and anyone from 60-100 was a senior.

We further used the function get_dummies() to convert necessary categorical data into dummy/indicator variables. We did this for the age category and the country category.

6.4 Feature Engineering

6.4.1 Number of Chronic Diseases

Although we had a binary column of whether or not a patient had a chronic disease or not, we decided that it would be interesting to see if the number of chronic diseases a person had correlated with their survival rate. Hence, we created a column to analyse this. Within this column, all blank or unknown values were populated with '-1' to indicate this was missing instead of a '0' which would indicate the person has no chronic diseases.

6.4.2 Hypertension, Diabetes, Heart diseases and Kidney diseases

From initial research that was conducted about COVID-19, the first few 26 patients that have succumbed to the novel CoV had significant health conditions including hypertension, diabetes, heart and/or kidney function issues that may have made them more susceptible [4]. Hence we decided to introduce 4 new columns for each disease to indicate if these diseases were present.

6.5 Dropping Rows and creating a Target Variable

After looking at all the features, it was time to finalise the target variable. Since we were looking at predicting whether or not the patient survived or not, we undoubtedly had to take the outcome column as our target variable or 'y'. There was a significant size of our dataset where the outcome was blank, and unfortunately we agreed to drop these rows since we required the target variable to be fully populated. The outcome column was a string-type column, hence we had to first identify which strings came under which category. Started off with four: death, critical, stable and recovered. Death/critical categories were then replaced with 0 and stable/recovered categories were replaced with 1.

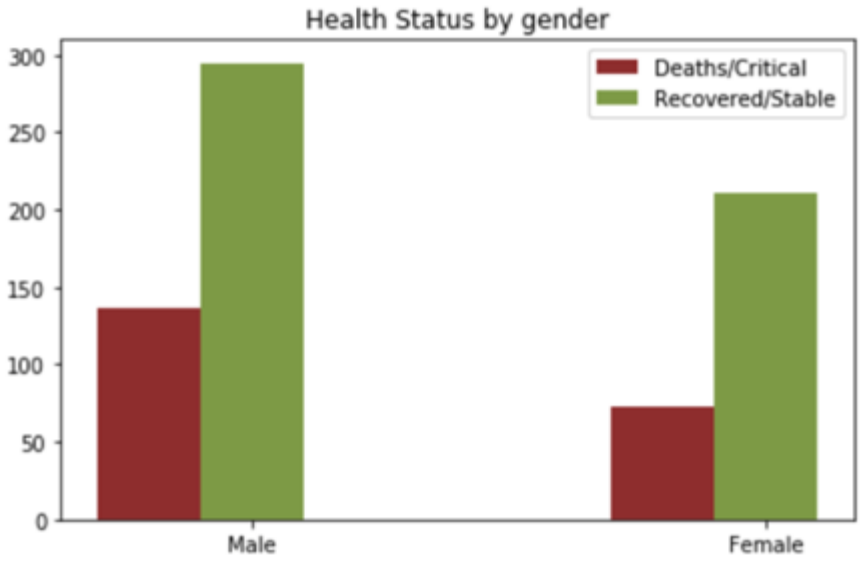
7 Data Exploration and Visualisation

The quality of our data decides the quality of our model's output. After the pre-processing of the data, we wanted to explore the data and get a better understanding. Specifically, we explored the existence of any outliers or missing values and their effect on our machine-learning models. In addition, it was also necessary to explore different correlations and patterns within the data to understand which features are most important and can support our machine-learning models. Exploration helped gain critical insights that we could have missed.

From our data, we wanted to understand the different factors that increase the chances of coronavirus patient mortality. From our research, we discovered that age, co-morbidity, and sex contributed to morbidity rates [9][10][12]. We used this as the starting point for our exploration. To explore the data we firstly queried the data to get all patients whose outcome was death/critical and recovered/stable. We used this data to plot all the following charts.

7.1 **Gender and Mortality**

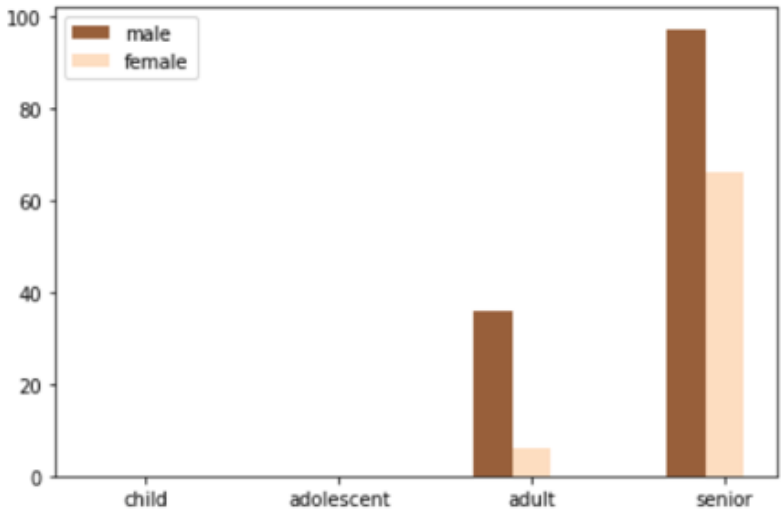
We wanted to observe the effects of gender on mortality of a COVID-19 patient. From the bar chart below, we see a comparison between female and male patients. The number of males in a critical state or dying is almost twice the females. Even for patients in stable/recovered state, the ratio of males is more than females. This shows there were more male COVID-19 cases that recovered. This reaffirms our research that gender does contribute to morbidity rates for a COVID-19 patient.



7.2 Age and mortality

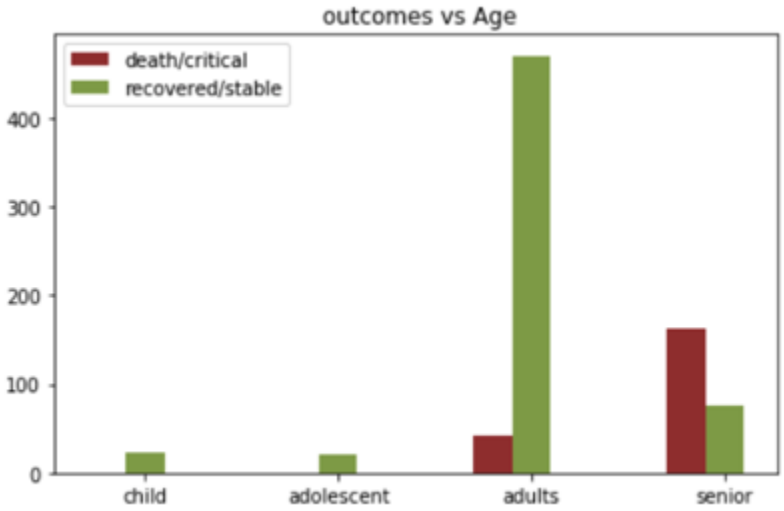
We wanted to explore the above results further by observing the effects of age along with gender. The chart below shows the age category along with gender versus the total number of critical/death outcomes. The results are consistent. Males' critical/death outcomes outweigh females'. For adult males, there are 70% more males' cases as compared to females. Even for seniors, there are almost 20% more males' cases. The results indicate gender does influence COVID-19 patient mortality and that males are more vulnerable than females.

Another interesting thing can be observed from the chart below. It can be seen that there are no critical/death cases for children or adolescents, only cases for adults and seniors. Therefore, age is another important factor. We explored age further by plotting outcomes for different age categories.



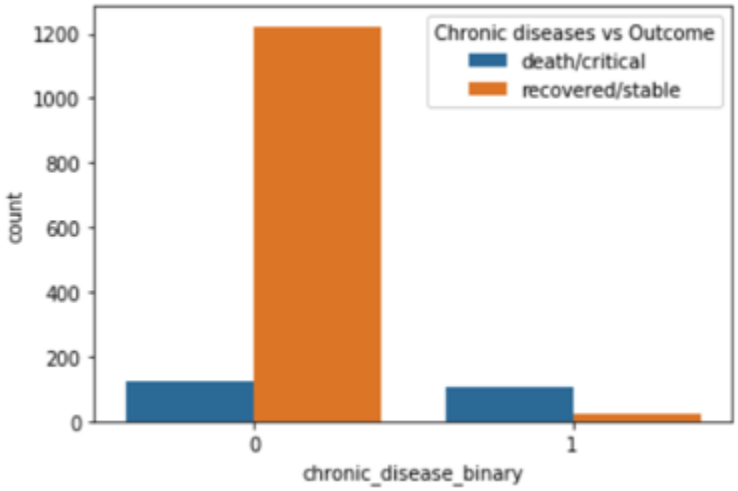
The bar chart below shows different age categories versus outcomes. It can be observed that there are only adults and senior death/critical cases. There are very few stable/recovered cases for children and adolescents but most of the cases are in the adult and senior age categories. This shows that in our data, there were no

children or adolescents that were in a critical state or died. For those that were infected, they recovered. This illustrates that age is an important factor for the mortality of COVID-19 patients and adults and seniors are the most vulnerable.



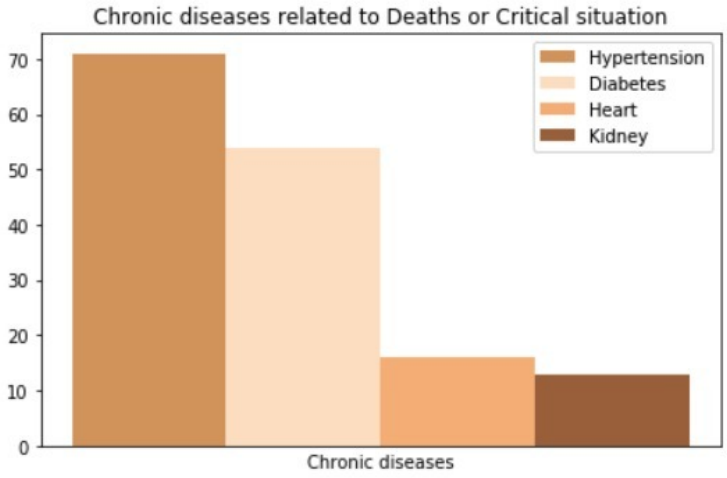
7.3 Co-morbidities and mortality

This part explores co-morbidities and how they affect mortality rates. The chart below shows patients with other chronic diseases and their outcomes. The chart doesn't indicate a strong relationship between other chronic diseases and outcomes. More patients with chronic diseases had critical/death as the outcome but the difference is insignificant. On the other hand, the majority of the patients whose outcome was recovered/stable did not have other chronic diseases. Intuitively, this is logical, as patients who do not have other diseases should be able to recover easily as compared to those patients who do have chronic diseases.



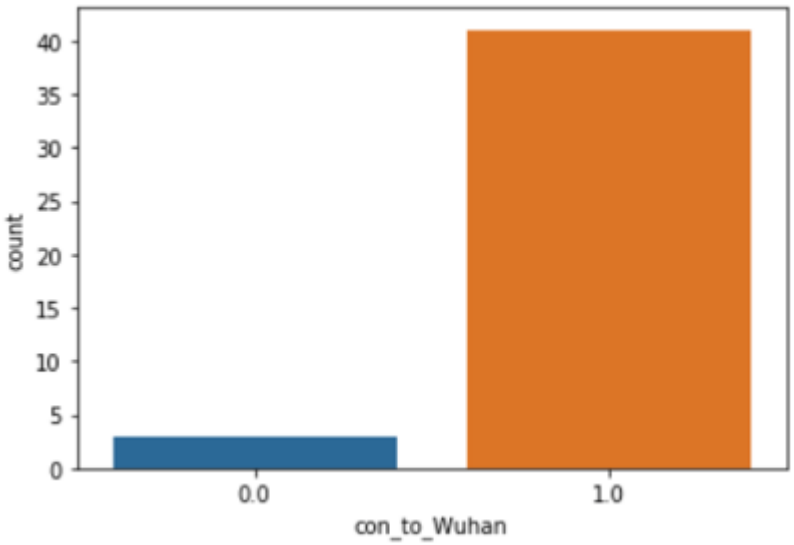
In further developing the chronic disease exploration, we investigated specific types of other chronic diseases patients suffered from. The chart below shows a comparison between different chronic diseases and the number of patients with critical/death as an outcome. The majority had hypertension and diabetes. Based on our research we were expecting the number of patients with heart and kidney diseases to be high as well [9,10], however, the graph reveals that that is not necessarily the case. A possible reason may be

due to our limited data or extensive null values. We concluded that although a relationship between chronic diseases and mortality rates exists, more data would enable us to validate such a relationship.



7.4 Connection to Wuhan

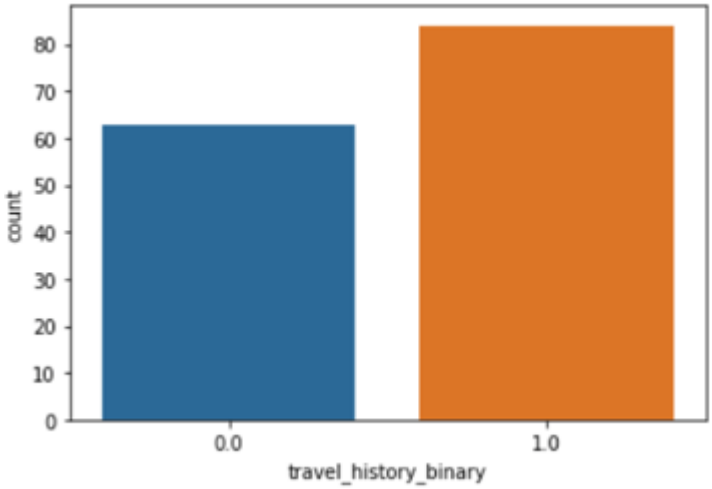
We know from research that COVID-19 started from Wuhan. We wanted to explore the connection of outcome with Wuhan. Results are no surprise, almost 94% of patients whose outcome was death/critical had some connection with Wuhan. This illustrates that connection to Wuhan is an important factor for mortality rates.



7.5 Travel history

Currently, due to the COVID-19 pandemic, we all are facing quarantine and are not allowed to travel. Therefore, we investigated the relation of travel history and the patients' outcomes. The below chart shows patient travel history and the number of patients whose outcome was death/critical. We initially thought travel history would be related to the patient's outcome because of the government's initiative regarding

lockdown and trying to flatten the death curve. But according to our results, there is not much difference between number death/critical cases for patients who travel and who didn't. This may be because travel history affects the spread of the virus not the patient's outcome.



8 Modelling

8.1 Feature Selection and Pre-Processing

For the feature selection, we simply took all the columns that were of type 'int', 'uint8' and 'float64'(i.e. numeric). We took the feature set as X, preparing it now for the modelling. We took the target variable which was the 'outcome' and assigned it to 'y'.

We normalised the feature set by using the logarithmic scale. Normalisation is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values, hence it can be very important.

We still had a significant chunk of our data in the feature set that was still blank or missing. Hence, our first model of choice was XGBoost classifier as it takes null values into consideration. However, for our second model, K-Nearest Neighbours (KNN) classifier, we decided to use KNN-impute to populate the missing values. This imputing method populates missing data by finding the k closest neighbors to the observation with missing data and then imputing them based on the non-missing values in the neighbours. We decided that this would be a more efficient way to fill in any null values instead of filling them with 0, because in the majority of the cases, '0' represented some piece of information.

8.2 Models

8.2.1 XGBoost Classifier

XGBoost is an open source library providing an implementation of gradient boosted decision trees. It is an algorithm that has recently been dominating applied machine learning because of its speed

and performance. It is a boosting ensemble method whereby decision trees are built sequentially such that each subsequent tree aims to reduce the errors of the previous tree.

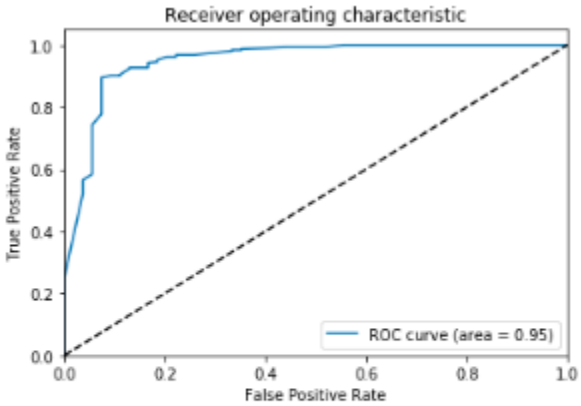
Why did we choose XGBoost?

One of the most influencing factors for choosing this classifier was because of its ability to handle missing data and an imbalanced dataset. This classifier has numerous different parameters.

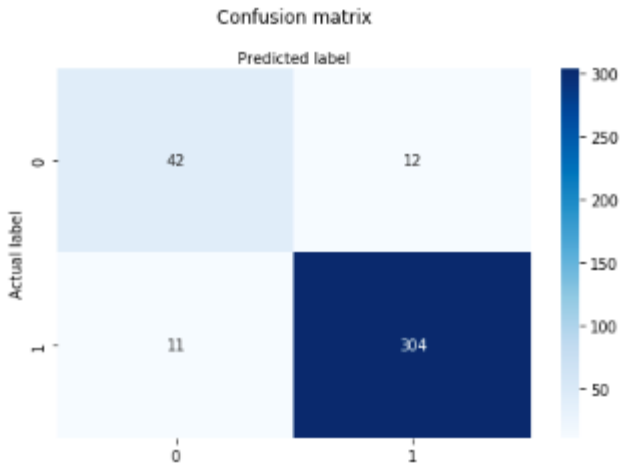
Our XGBoost Model implementation:
To implement the XGBoost algorithm, we took the following steps

1. We used the pre-processed data and gathered all the numerical values.
2. The pre-processed data left us with X, which was the feature set and y as the target variable.
3. Normalise data by using log and max-min methods.
4. We left the missing values as they are
5. Split the data into 25% train and 75% test data.
6. Use train data to train our model and validated the parametres by hypertuning and cross-validation through GridSearchCV
7. Then used the trained model to predict test data.
8. Lastly, we used the predictions to calculate the accuracy, precision, recall, f1 score, AUC score, and confusion matrix of our model. We use these to assess the quality of our models.

The Model Results:
We used the ROC curve, along with the confusion matrix to analyse the results of our model.

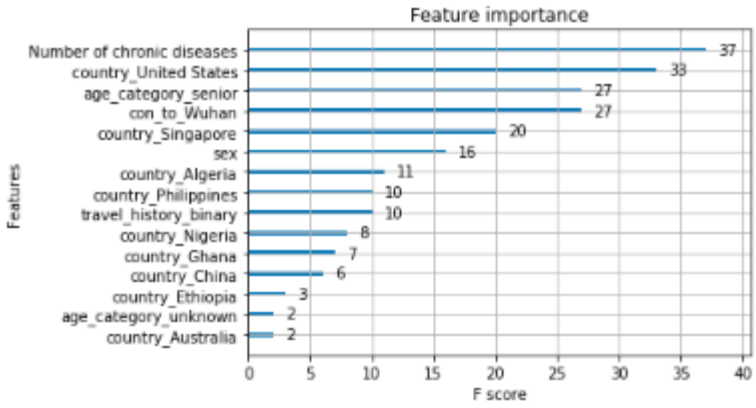



```
AUC 0.9495590828924162
Accuracy 0.9376693766937669
Precision 0.9620253164556962
Recall 0.9650793650793651
F1 0.9635499207606973
```



Classification Report:

	precision	recall	f1-score	support
0.0	0.79	0.78	0.79	54
1.0	0.96	0.97	0.96	315
micro avg	0.94	0.94	0.94	369
macro avg	0.88	0.87	0.87	369
weighted avg	0.94	0.94	0.94	369



According to the confusion matrix, the model was able to predict 96% of stable/recovered labels as compared to 79% of critical/death. This may be because there was more data related to stable/recovered outcomes in the dataset and the model was able to train better for that outcome, or this could possibly be a case of overfitting as well. The death rate of COVID-19 is not significantly high which explains why the critical/death class is significantly smaller than the stable/recovered label

Overall there is 94% accuracy, 96% precision and 97% recall. ROC area is 95%, showing the amount of data the model was correctly able to classify. These results demonstrate that the model performed extremely well. Especially considering the limited amount of data we used to train the model.

The model also showed the Feature Importance of the model. The F1 Score was used to measure the importance which is based on both the Recall and Precision. These are the two standard scores

usually used. Recall and Precision are known to be misleading when reported independently, which is why the F1 score offers the harmonic mean between the two [19].

8.2.2 KNN Classifier

1. KNN is a supervised machine learning algorithm that can also be used for both classification and regression predictive problems. This algorithm works on the concept of proximity. It assumes that similar things exist near each other. The algorithm calculates the distance between different points, then it classifies them based on their distances from its K nearest neighbours.

Pseudocode:

1. Load data
2. Chose the K value (number of nearest neighbour)
3. For each of data point
    3.1 Calculate Euclidean distance to all data points. Beside Euclidean, other methods can be used to calculate the distance such as Hamming or Manhattan
    3.2 Store the distances in an ordered collection
4. Get the first K entries from the collections
5. Get the labels of the selected K entries
    5.1 If used for classification, it returns the mode of K labels.
    5.2 If used for regressing it returns the mean of the K labels.

Why did we choose KNN?

KNN is very simple and easy to implement. In KNN there is no need to build models and tune several parameters. It is a non-parametric algorithm as it doesn't make any assumption on the underlying data distribution. This was useful for us as we didn't have any prior knowledge about data distribution related to COVID-19 patients.

Besides being a simple algorithm KNN is useful in solving problems that depend on identifying similar objects such as a recommender system. In our project, we are trying to find similar features of COVID-19 patients who had the same outcome. Thus, helping us to predict the outcome of a new COVID-19 patient.

Our KNN implementation:

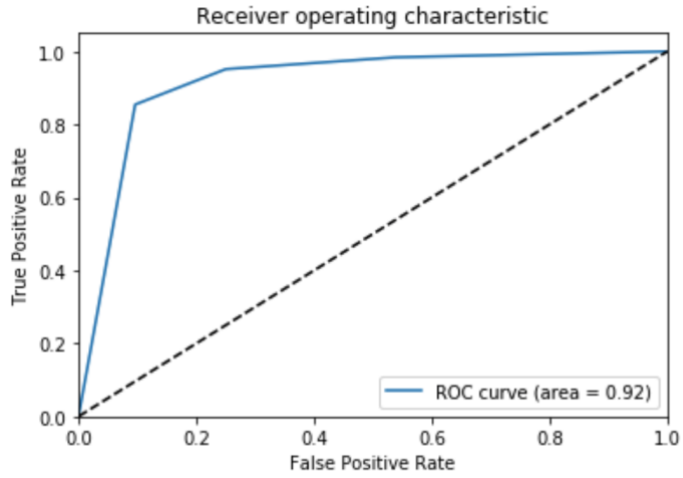
To implement KNN for our data, we took the following steps

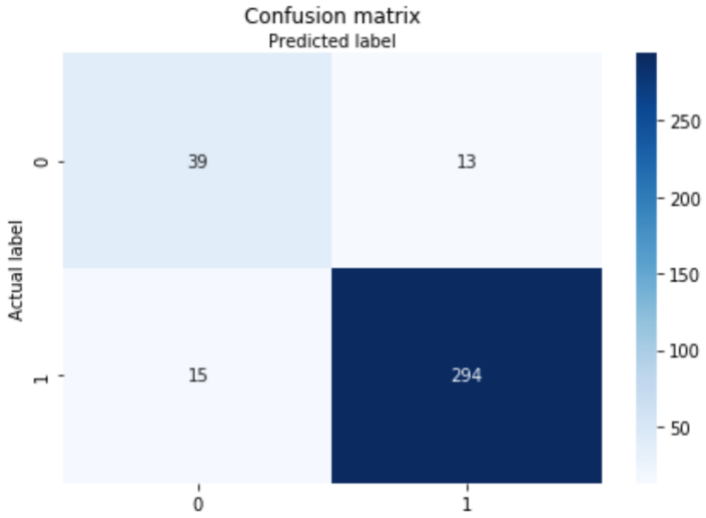
1. We used the pre-processed data and gathered all the numerical values.
2. We used all the columns except the outcome as our predictors. We used the outcome column as our labels.
3. Normalise data by using log and max-min methods.
4. We filled missing data using KNN impute
5. Split the data into 25% train and 75% test data.
6. Use train data to train our model
7. Then used the trained model to predict test data.

8. Lastly, we used the predictions to calculate the accuracy, precision, recall, f1 score, AUC score, and confusion matrix of our model. We use these to assess the quality of our models.

The Model Results:

We have used the confusion matrix and ROC curved to understand the results and performance of our models.





AUC 0.9169156086631813
Accuracy 0.9224376731301939
Precision 0.9576547231270358
Recall 0.9514563106796117
F1 0.9545454545454545

According to the confusion matrix, the model was able to predict 95% of stable/recovered labels as compared to 75% of critical/death. This may be because there was more data related to stable/recovered outcomes in the dataset and the model was able to train better for that outcome.

But overall there is 92% accuracy. and 95% F1 score. ROC area is 92%, showing the amount of data the model was correctly able to classify. These results demonstrate that the model performed quite well. Especially considering the limited amount of data we used to train the model.

8.3 Comparing Models

When we look at comparing models, we can very evidently see that there is not much significant difference between the results. The XGBoost classifier performed slightly better than KNN.

This could be due to the fact that XGBoost has multiple different parameters and different combinations to fit the model in. It takes several different considerations into account, which KNN does not due to its limited number of parameters.

Both these models were chosen since no assumptions on distribution of data is actually required. Nonetheless it is important to note, that XGBoost and other decision tree models are more prone to overfitting.

9 Successes and Challenges

Our successes were immense. We achieved far better results than expected, and worked well together as a team. Most importantly, our results from the features importance of the XGBoost classifier resonated with our research.

However, on the other hand, multiple challenges were faced during the research. These challenges predominately were based on the massive amounts of data that was missing. It was a compromise on where and how to get the data due to the scope of research being done within the domain. We had to weigh out whether or not we wanted a fully populated dataset- which would not have majority of the data required to produce this outcome, or have a sparser dataset, with more features to investigate. We all concluded that the later would potentially be better. Another challenge was the time-frame. Initially, it was agreed upon that due to the limited data (due to time frame of existence for COVID), we would compare SARS data for the same time period (approximately 4 months), and understand if we could make similar conclusions to if we had data for a longer period of time. This would reassure that the COVID data that we obtained, may be limited but were justifiable and valid. However, this could not be implemented.

10 Possible extensions

As COVID-19 is an unknown virus, every day new information is emerging, and data is changing. Our report is limited to the current knowledge we have about COVID-19. More information about COVID-19 can greatly enhance our analysis and findings. Such as can a COVID-19 patient's genetics or race play a role in their survival. More material on COVID-19 may lead to choosing different models and different strategies for our analysis thus the possibility of extension.

Furthermore, during the writing of this report, we were limited by the quantity and quality of data. We had a limited amount of data that had many missing values. All of these factors affect our results. For example, if we had a patient occupation column properly populated, we could have used it to enhance our models and our findings. More quality data can be used to find new correlations within data, improving the models, and may lead to new our findings.

11 Concluding Remarks

We achieved the aim of the project and learnt valuable insight into the current epidemic. Our focus was on whether or not we could predict the survival of a COVID patient and relate the feature importance back to significant features highlighted in other papers and we were successful.

As far as the data analysis process is concerned, we learned that missing data is a real world challenge which presents more than one issue and can have an effect on multiple different factors within your data.

We started our data exploration based on this literature review and looked into features such as age, sex, co-morbidity, travel history and connection to Wuhan. They showed correlations with mortality which were consistent with the hypothesis.

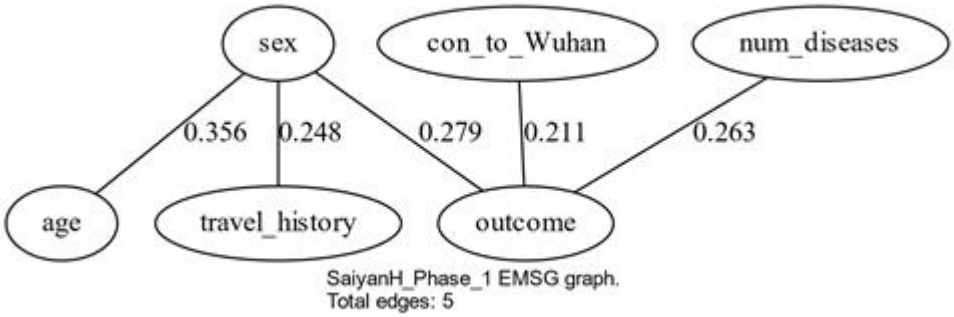
Both models, XGBoost and K-Nearest Neighbours, produced exceptional results. The Feature Importance produced outcomes in what the group hoped to expect. Being of a higher age, connection to Wuhan and number of chronic diseases played in the league for top five features.

12 Bayesian Network Structure 2 Learning Analysis

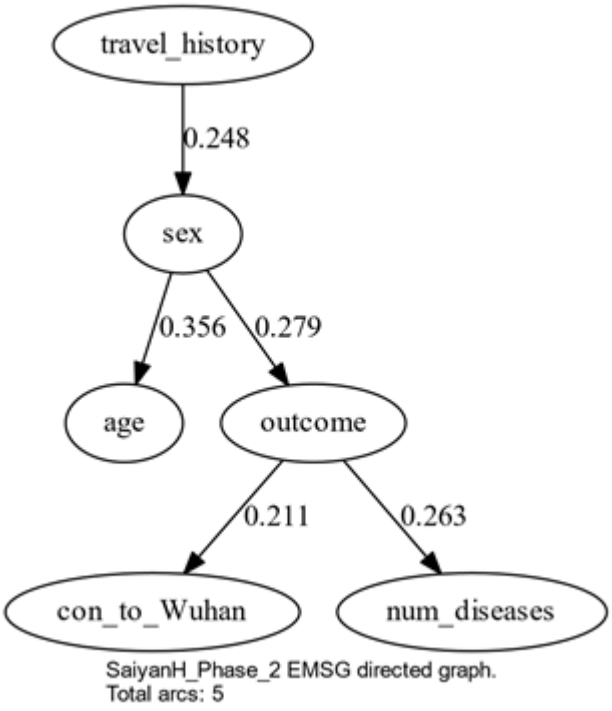
Step 3 Knowledge Graph

_____ Evaluation _____
Nodes: 6
Sample size: 1473
TrueDAG arcs: 6
TrueDAG independencies: 9
LearnedDAG arcs: 6
LearnedDAG independencies: 9
_____ Stats from metrics and scoring functions _____
# of independent graphical fragments: 1
_____ Inference-based evaluation _____
BIC/MDL score -9671.756
# of free parameters 171
BUILD SUCCESSFUL (total time: 20 seconds)

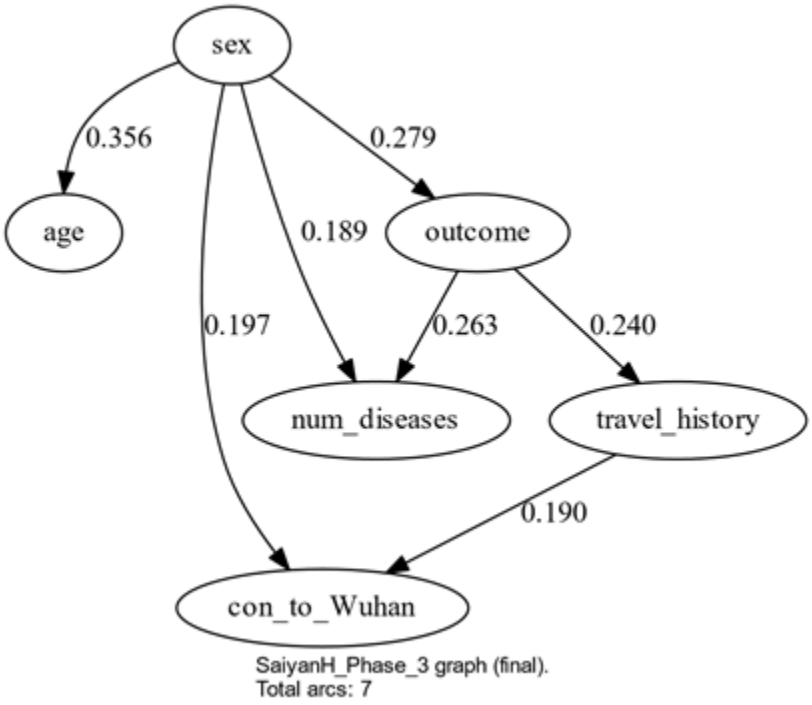
SaiyanH Phase 1



SaiyanH Phase 2



SaiyanH Phase 3



Step 4 Output

_____ Stats from metrics and scoring functions _____
Precision score: 0.357
Recall score: 0.417
F1 score: 0.385
SHD score: 7.500
DDM score: -0.833
BSF score: -0.028
# of independent graphical fragments: 1

_____ Inference-based evaluation _____
BIC/MDL score -7864.781
# of free parameters 85
BUILD SUCCESSFUL (total time: 16 seconds)

Question 1: Describe the steps you followed to produce the knowledge-based causal graph. For example, whose and what knowledge did you use? If knowledge was elicited by multiple members of the group, how did you handle disagreements?

An initial presentation of a knowledge-based graph was drawn, which led further discussion. The basis of the graph stemmed from the feature importance graph produced by the XGBoost classifier, along with the research papers that were read. After conversation by all members, it was agreed unanimously to remove country variables as there were too many, and the binary chronic disease variable. We included age, number of chronic diseases, connection to Wuhan, travel history, sex, and outcome.

There was a debate over the suggestion to split the outcome into its 2 states, after which the idea was abandoned as no impasse could be made. Overall, we managed to handle the debate very sensibly and everyone was happy on the final result.

Question 2: In your own words, explain the difference between the Phase_1 and Phase_2 graphs. How did the algorithm produce the Phase 2 graph from the Phase_1 graph?

During Phase_1, the associational learning uses discrepancies between prior and posterior marginal probabilities. An undirected graph is used which starts with an initial best guess. In Phase_2, the results from Phase_1 are taken as a starting point and iterations of constraint-based pruning in triples takes place.

During Phase_1 MMD scores are used to produce initial best guess undirected graphs. MMD is calculated by using discrepancy between prior and posterior probabilities. This graph preserved more than one connecting path from one node to another thus giving a denser starting undirected graph. Phase_2 uses the result from Phase_1 and checks for dependence or independence. It applies conditional constraints to classify each node into conditional dependence, independence or insignificance.

Question 3: In your own words, explain the difference between the Phase_2 and Phase_3 graphs. How did the algorithm produce the Phase_3 graph from the Phase_2 graph? If both graphs are identical, how do you explain this result?

Phase_2 performs a test to find conditional dependence, independence, or insignificance. Phase_3 uses a graph from Phase_2 to evaluate the graph based on neighbouring graph exploration and scoring criterion. This phase uses BIC to score the graphs. It uses the Hill climbing method to explore the graph and to avoid local minima, this phase uses Tabu search. Our graphs from both phases were different. If they had been identical, it would be because there was no further improvement found in Phase_3.

Question 4: List the number of scores generated in each CSV file. For example, if marginalDep.csv has 100 rows of scores, then you should write '100' for that particular file. Discuss the different quantities in scores generated in each file. For example, why do you think there are 100 scores in the file marginalDep.csv and 200 scores in the file conditionalInsignificance.csv?

- Conditional insignificance: 58
- Conditional Dependence: 1
- Conditional independence: 1
- Marginal Dependence: 15

Marginal dependencies are lower because they are processed in the Phase_1 and involve less constraints, whereas conditional insignificance is higher because they are nodes are classified in Phase_2 and Phase_3, where more constraints are applied, invoking strict checking for classification into categories. When the program is no longer able to locate conditional dependencies and interdependencies, it classifies those links into conditional insignificance.

Question 5: Refer to your F1, SHD and BSF scores and compare them to the related scores shown in Fig 2 of the related research paper. Are your scores mostly lower, on par, or higher (in general) compared to those shown in Fig 2 with respect to SaiyanH (ignore results from other algorithms)? Indicate whether this result is in agreement or not with your initial expectations, and explain why.

Across all three statistics, our SaiyanH graphs show poor accuracy when compared to most of the case studies' accuracies of the same approximate sample size of 1K. This may be due to the vast amount of missing data that is prevalent in the dataset, or the low number of both nodes and links. Magnifying into the F1 of .385, SHD of 7.5 and BSF of -0.028, these are only consistent with only 1/6 of the case studies (BSF of close to zero means that most direct independencies have been found while few direct dependencies were found). This result was expected due to the issue of missing data.

Question 6: Refer to your elapsed time of structure learning and compare it to the runtime shown in Table 2 of the related research paper. Indicate whether your result is consistent or not with the results shown in Table 2, and explain why.

When compared to the 8 node cases of the same 1K sample size, which ran for one second, our structured learning (6 nodes) ran for 16 seconds. This discrepancy could be explained by the fact that score-based learning (Phase Three) is the algorithm's most time-consuming phase of the algorithm for low sample sizes.

Question 7: Compare the BIC/MDL score generated at Step 4 with the BIC/MDL score generated at Step 3. What do you understand from the difference in those two scores? Indicate whether this result is in agreement with your initial expectations, and explain why.

The BIC/MDL score generated at Step 4 was lower than the score generated at Step 3 (Step 3 = -9671.756 vs Step 4 = -7864.781) because during Step 4, structure learning involved both the constraint-based pruning process and the escape the local maximum by score-based pruning process (iterative improvements in the BIC score by TABU searching). Step 3 was a simple algorithm producing an undirected graph with simple probabilities. This result was expected due to the process of the structure learning in the SaiyanH program.

Question 8: Compare the # of free parameters generated at Step 4 with the # of free parameters generated at Step 3. What do you understand from the difference between these two values? Indicate whether this result is in agreement with your initial expectations, and explain why.

During Step 4's structure learning, different parameters are tested to reach different BIC/MDL results over the search space, and the parameters that failed to add value to the process are removed. This supports the difference (reduction) between Step 3 and Step 4's free parameters of 171 and 85 respectively, and it aligns with our initial expectations.

13 References

[1] https://coronavirus.jhu.edu/ accessed on April 30, 2020, 6:08 a.m.

[2] https://www.medicinenet.com/script/main/art.asp?articlekey=228841 accessed on April 30,2020, 3:12 a.m.

[3] Y. Yang, M. Islam, J. Wang, Y. Li and X. Chen, "Traditional Chinese Medicine in the Treatment of Patients Infected with 2019-New Coronavirus (SARS-CoV-2): A Review and Perspective", *International Journal of Biological Sciences*, vol. 16, no. 10, pp. 1708-1717, 2020. Available: http://www.ijbs.com/v16p1708.htm. [Accessed 30 April 2020].

[4] L. Yan et al., "A machine learning-based model for survival prediction in patients with severe COVID-19 infection", 2020. Available: 10.1101/2020.02.27.20028027 [Accessed 30 April 2020].

[5] Z. Sun, K. Thilakavathy, S. Kumar, G. He and S. Liu, "Potential Factors Influencing Repeated SARS Outbreaks in China", *International Journal of Environmental Research and Public Health*, vol. 17, no. 5, p. 1633, 2020. Available: 10.3390/ijerph17051633.

[6] N. Jia et al., "Case fatality of SARS in mainland China and associated risk factors", *Tropical Medicine & International Health*, vol. 14, pp. 21-27, 2009. Available: 10.1111/j.1365-3156.2008.02147.x [Accessed 30 April 2020].

[7] https://www.bbc.co.uk/news/world-51322733 accessed on April 30, 2020, 6:48 a.m.

[8] J. Wu et al., "Estimating clinical severity of COVID-19 from the transmission dynamics in Wuhan, China", *Nature Medicine*, vol. 26, no. 4, pp. 506-510, 2020. Available: 10.1038/s41591-020-0822-7 [Accessed 30 April 2020].

[9] L. Gralinski and V. Menachery, "Return of the Coronavirus: 2019-nCoV", *Viruses*, vol. 12, no. 2, p. 135, 2020. Available: 10.3390/v12020135 [Accessed 30 April 2020].

[10] H. Chang et al., "Hematological and Biochemical Factors Predicting SARS Fatality in Taiwan", *Journal of the Formosan Medical Association*, vol. 105, no. 6, pp. 439-450, 2006. Available: 10.1016/s0929-6646(09)60183-2.

[11] F. Di Gennaro et al., "Coronavirus Diseases (COVID-19) Current Status and Future Perspectives: A Narrative Review", *International Journal of Environmental Research and Public Health*, vol. 17, no. 8, p. 2690, 2020. Available: 10.3390/ijerph17082690 [Accessed 1 May 2020].

[12] C. Wu et al., "Risk Factors Associated With Acute Respiratory Distress Syndrome and Death in Patients With Coronavirus Disease 2019 Pneumonia in Wuhan, China", *JAMA Internal Medicine*, 2020. Available: 10.1001/jamainternmed.2020.0994.

[13] F. Zhou et al., "Clinical course and risk factors for mortality of adult inpatients with COVID-19 in Wuhan, China: a retrospective cohort study", *The Lancet*, vol. 395, no. 10229, pp. 1054-1062, 2020. Available: 10.1016/s0140-6736(20)30566-3.

[14] W. Han et al., "The course of clinical diagnosis and treatment of a case infected with coronavirus disease 2019", *Journal of Medical Virology*, vol. 92, no. 5, pp. 461-463, 2020. Available: 10.1002/jmv.25711.

[15] Y. Cao et al., "Comparative genetic analysis of the novel coronavirus (2019-nCoV/SARS-CoV-2) receptor ACE2 in different populations", *Cell Discovery*, vol. 6, no. 1, 2020. Available: 10.1038/s41421-020-0147-1.

[16] X. He et al., "Temporal dynamics in viral shedding and transmissibility of COVID-19", *Nature Medicine*, 2020. Available: 10.1038/s41591-020-0869-5.

[17] M. Sajadi, P. Habibzadeh, A. Vintzileos, S. Shokouhi, F. Miralles-Wilhelm and A. Amoroso, "Temperature and Latitude Analysis to Predict Potential Spread and Seasonality for COVID-19", *SSRN Electronic Journal*, 2020. Available: 10.2139/ssrn.3550308.

[18] Who.int, 2020. [Online]. Available: https://www.who.int/csr/sars/en/WHOconsensus.pdf. [Accessed: 13- May- 2020].

[19] A. Constantinou, "Learning Bayesian Networks that enable full propagation of evidence", arXiv.org, 2020. [Online]. Available: https://arxiv.org/abs/2004.04571. [Accessed: 10- May- 2020].

[20]"beoutbreakprepared/nCoV2019",GitHub,2020.[Online].Available: https://github.com/beoutbreakprepared/nCoV2019. [Accessed: 13- April- 2020].

14 Appendix

a. The Python Script:

```
import numpy as np

import scipy as sp

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

import datetime as dt

warnings.filterwarnings("ignore")

get_ipython().run_line_magic('matplotlib', 'inline')

from sklearn.preprocessing import StandardScaler

import re


from sklearn.model_selection import train_test_split

import xgboost as xgb

from sklearn.model_selection import KFold

from sklearn.model_selection import cross_val_score

from sklearn.metrics import roc_curve, auc, precision_recall_curve, accuracy_score,
confusion_matrix, classification_report, fbeta_score,  r2_score


pd.set_option('display.max_columns', 50)

pd.set_option('display.max_rows', 100)


from sklearn.model_selection import (GridSearchCV,

                                     train_test_split)

from sklearn.metrics import (classification_report)

from matplotlib import pyplot as plt

import xgboost as xgb
```

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
get_ipython().run_line_magic('matplotlib', 'inline')
from sklearn import manifold


def test_results(y_test, y_test_pred, y_test_pred_pr):

    print("**********TEST RESULTS**********")

    auc_plotting(y_test, y_test_pred_pr)
    auc_score, accuracy, precision, recall, f1 = results_summary(y_test, y_test_pred, y_test_pred_pr)
    print("Classification Report:")
    print(sklearn.metrics.classification_report(y_test, y_test_pred))
    return auc_score, accuracy, precision, recall, f1


def auc_plotting(y_test, y_pred_probs):
    fpr, tpr, thresholds = roc_curve(y_test, y_pred_probs)
    roc_auc = auc(fpr, tpr) # compute area under the curve
    plt.figure()
    plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % (roc_auc))
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")
    plt.show()
```

```
def results_summary(y_test,y_pred, y_test_pred_pr):
    auc_score = sklearn.metrics.roc_auc_score(y_test, y_test_pred_pr)
    accuracy = sklearn.metrics.accuracy_score(y_test, y_pred)
    precision = sklearn.metrics.precision_score(y_test, y_pred)
    recall = sklearn.metrics.recall_score(y_test, y_pred)
    kappa = sklearn.metrics.cohen_kappa_score(y_test, y_pred)
    f1 = sklearn.metrics.f1_score(y_test, y_pred)
    print("AUC %s " % auc_score)
    print("Accuracy %s " % accuracy)
    print("Precision %s " % precision)
    print("Recall %s " % recall )
    print("F1 %s " % f1)



    print("Confusion Matrix:")
    # Confusion Matrix for evaluaiton of results
    cnf_matrix_log = confusion_matrix(y_test, y_pred)
    class_names=['Active','Churned'] # name  of classes
    fig, ax = plt.subplots()
    tick_marks = np.arange(len(class_names))
    plt.xticks(tick_marks, class_names)
    plt.yticks(tick_marks, class_names)
    # create heatmap
    sns.heatmap(pd.DataFrame(cnf_matrix_log), annot=True, cmap="Blues" ,fmt='g')
    ax.xaxis.set_label_position("top")
    plt.tight_layout()
    plt.title('Confusion matrix', y=1.1)
    plt.ylabel('Actual label')
    plt.xlabel('Predicted label')
```

```
    plt.show()
    return auc_score, accuracy, precision, recall, f1



# data = pd.read_csv('https://raw.githubusercontent.com/beoutbreakprepared/nCoV2019/master/latest_data/latestdata.csv', encoding = 'latin',  index_col = False)
data = pd.read_csv('nCoV2019/latest_data/latestdata.csv', encoding = 'Latin')


data.shape


# # Pre-processing
df = data.drop(columns=['longitude', 'latitude', 'geo_resolution', 'source', 'notes_for_discussion',
                        'admin_id', 'admin1', 'admin2', 'admin3', 'data_moderator_initials',
                        'sequence_available', 'city', 'province'])
df = df.rename(columns={"lives_in_Wuhan": "con_to_Wuhan"})


df['sex'] = df['sex'].replace('female', 1)
df['sex'] = df['sex'].replace('male', 0)


df['con_to_Wuhan'] = df['con_to_Wuhan'].replace('yes', 1)
df['con_to_Wuhan'] = df['con_to_Wuhan'].replace('no', 0)
df['con_to_Wuhan'] = df['con_to_Wuhan'].replace('no, work in Wuhan', 1)


df['travel_history_binary'] = df['travel_history_binary'].astype(str)
df['chronic_disease_binary'] = df['chronic_disease_binary'].astype(str)


df['travel_history_binary'] = df['travel_history_binary'].replace('True', 1)
df['travel_history_binary'] = df['travel_history_binary'].replace('False', 0)


df['chronic_disease_binary'] = df['chronic_disease_binary'].replace('False', 0)
```

```
df['chronic_disease_binary'] = df['chronic_disease_binary'].replace('True', 1)
df['travel_history_binary'] = df['travel_history_binary'].replace('nan', np.nan)
df = df.set_index('ID')

df.isnull().sum().sort_values(ascending = False).plot(kind='bar')

df['chronic_disease'] = df['chronic_disease'].replace('nan', -1)
df['chronic_disease'] = df['chronic_disease'].replace(r'^\s*$', -1, regex=True)
df['chronic_disease'] = df['chronic_disease'].replace(np.nan, -1)

delimiters = ";", ",", ":"
no_chronic_disease = []
regexPattern = '|'.join(map(re.escape, delimiters))
for index, row in df.iterrows():
    if( row.chronic_disease is -1 or 'http' in row.chronic_disease):
        no_chronic_disease.append(-1)
    else:
        no_chronic_disease.append(len(re.split(regexPattern, row.chronic_disease)))
df['Number of chronic diseases'] =  no_chronic_disease
df['Number of chronic diseases'].value_counts()

df['Hypertension'] = df['chronic_disease'].str.contains('hypertension|hypertensive')
df['Hypertension'] = df['Hypertension'].apply(lambda x: 1 if x == True else 0)

df['Diabetes'] = df['chronic_disease'].str.contains('diabetes')
df['Diabetes'] = df['Diabetes'].apply(lambda x: 1 if x == True else 0)

df['Heart'] = df['chronic_disease'].str.contains('heart|coronary|bypass|cardiac|cardio')
df['Heart'] = df['Heart'].apply(lambda x: 1 if x == True else 0)
```

```
df['Kidney'] = df['chronic_disease'].str.contains('kidney|Kidney')

df['Kidney'] = df['Kidney'].apply(lambda x: 1 if x == True else 0)


df.drop(columns = ['chronic_disease'], inplace = True)

df = pd.get_dummies(df, columns=['country'])


df['outcome'] = df['outcome'].replace(['died', 'Death', 'Deceased', 'Died', 'dead', 'death', 'Dead'],
'Died')

df['outcome'] = df['outcome'].replace(['Critical condition', 'critical condition', 'severe illness',
'severe',  'critical condition, intubated as of 14.02.2020', 'unstable'], 'Critical')

df['outcome'] = df['outcome'].replace(['recovered', 'Recovered', 'recovering at home 03.03.2020',
'not hospitalized', 'Discharged', 'Discharged from hospital', 'released from quarantine', 'discharge',
'discharged', 'Alive'],'Recovered')

df['outcome'] = df['outcome'].replace(['stable', 'stable condition', 'Stable', 'Under treatment',
'Symptoms only improved with cough. Currently hospitalized for follow-up.', 'Receiving
Treatment', 'treated in an intensive care unit (14.02.2020)'], 'Stable')


df = df[~df.outcome.str.contains("http", na = False)]

df = df[~df.outcome.str.contains("gov", na = False)]

df = df[~df.outcome.str.contains("State", na = False)]


df['outcome'] = df['outcome'].replace(['Stable', 'Recovered'], 1)

df['outcome'] = df['outcome'].replace(['Died', 'Critical'], 0)

df = df.dropna(subset =['outcome'])



# Cleaning age column

#function to return age categories

def bucket_age(age):

    if(age == -1):

        return 'unknown'

    if(age <= 9):
```

```
        return 'child'
    if(age <=19):
        return 'adolescent'
    if(age <=59):
        return 'adult'
    if (age <=100):
        return 'senior'
    else:
        return 'unknown'



# replacing null values
df['age'] = df['age'].replace('nan', -1)
# creating new column
age_category = []
for index, row in df.iterrows():
    if(isinstance(row.age, float)):
         age_category.append(bucket_age(row.age))
    elif( row.age is -1 ):
        age_category.append(bucket_age(-1))

    elif('.' in row.age):
        age = (float(row.age))
        age_category.append(bucket_age(age))
    elif('-' in row.age):
        ages= (row.age).split('-')
        if(ages[0] and ages[1]):
            age1= (float(ages[0]))
            age2= (float(ages[1]))
            mean = (age1 + age2) / 2
```

```
        else:
            mean= (float(ages[0]))
        age_category.append(bucket_age(mean))
        elif('months' in row.age or 'month' in row.age or 'weeks' in row.age):
        age_category.append(bucket_age(0))
    elif(len(row.age) == 4):
        age_category.append(bucket_age(-1))
    else:
        age_category.append(bucket_age(int(row.age))
df['age_category'] = age_category
df = pd.get_dummies(df, columns=['age_category'])

# # Visualisation
#  plot correlation's matrix to explore dependency between features

def plot_correlation(data):
    fig = plt.figure(figsize=(15, 15))
    sns.heatmap(data.corr(), annot=True, fmt=".2f")
    plt.show()

# plot correlation & densities
plot_correlation(df[['outcome','sex','con_to_Wuhan','age_category_adult','age_category_adolescent'
,'age_category_child','age_category_senior',
                'Number of chronic diseases','Hypertension','Diabetes','Heart', 'Kidney']])

# Check correlation between death and sex

#Get all data where outcome was death
deaths_critical = df.query('outcome == "0.0"')
recovered_stable = df.query('outcome == "1.0"')
```

```
sns.countplot(data=deaths_critical, x = 'sex')


sns.countplot(data=deaths_critical, x = 'travel_history_binary')
sns.countplot(data=deaths_critical, x = 'con_to_Wuhan')
chronic_diesease = sns.countplot(data=df, x = 'chronic_disease_binary', hue='outcome')
plt.legend(title='Chronic diseases vs utcome', labels=['death/critical', 'recovered/stable'])
plt.show(chronic_diesease)

# Different chronic diseases correlation to patient death or critical situation

Hypertension = len((deaths_critical.query('Hypertension == "1"')).axes[0])
Diabetes =len((deaths_critical.query('Diabetes == "1"')).axes[0])
Heart = len((deaths_critical.query('Heart == "1"')).axes[0])
Kidney = len((deaths_critical.query('Kidney == "1" ')).axes[0])
Hypertension = (Hypertension)
Diabetes = (Diabetes)
Heart = (Heart)
Kidney = (Kidney)

fig, ax = plt.subplots()
index = np.arange(1)
bar_width = 0.2
opacity = 0.8

rects1 = plt.bar(index - bar_width, Hypertension, bar_width,
alpha=opacity,
color='peru',
label='Hypertension')

rects2 = plt.bar(index, Diabetes, bar_width,
```

```
alpha=opacity,
color='peachpuff',
label='Diabetes')

rects3 = plt.bar(index + bar_width, Heart, bar_width,
alpha=opacity,
color='sandybrown',
label='Heart')

rects4 = plt.bar(index + (2*bar_width) , Kidney, bar_width,
alpha=opacity,
color='saddlebrown',
label='Kidney')

plt.title('Chronic diseases related to Deaths or Critical situation')
plt.xticks([])
plt.legend()
plt.xlabel('Chronic diseases')
plt.tight_layout()
plt.show()

male_death_critical = len((deaths_critical.query('sex == "0"')).axes[0])
female_deaths_critical =len((deaths_critical.query('sex == "1"')).axes[0])

male_recovered_stable = len((recovered_stable.query('sex == "0"')).axes[0])
female_recovered_stable =len((recovered_stable.query('sex == "1"')).axes[0])

objects = ('Male', 'Female')
deaths_critical = (male_death_critical, female_deaths_critical)
recovered_stable = (male_recovered_stable, female_recovered_stable)
```

```
fig, ax = plt.subplots()
index = np.arange(2)
bar_width = 0.2
opacity = 0.8

rects1 = plt.bar(index - bar_width, deaths_critical, bar_width,
alpha=opacity,
color='maroon',
label='Deaths/Critical')

rects2 = plt.bar(index, recovered_stable, bar_width,
alpha=opacity,
color='olivedrab',
label='Recovered/Stable')

plt.title('Health Status by gender')
plt.xticks(index , objects)
plt.legend()

plt.tight_layout()
plt.show()

adolescent_death_critical = len((deaths_critical.query('age_category_adolescent == "1"')).axes[0])
adult_deaths_critical =len((deaths_critical.query('age_category_adult == "1"')).axes[0])
child_deaths_critical = len((deaths_critical.query('age_category_child == "1"')).axes[0])
senior_deaths_critical =len((deaths_critical.query('age_category_senior == "1"')).axes[0])
adolescent_recoverd_stable = len((recovered_stable.query('age_category_adolescent == "1"')).axes[0])
adult_recoverd_stable =len((recovered_stable.query('age_category_adult == "1"')).axes[0])
```

```
child_recoverd_stable= len((recovered_stable.query('age_category_child == "1"')).axes[0])

senior_recoverd_stable =len((recovered_stable.query('age_category_senior == "1"')).axes[0])

deaths_critical = (child_deaths_critical,
adolescent_death_critical,adult_deaths_critical,senior_deaths_critical)

recovered_stable = (child_recoverd_stable, adolescent_recoverd_stable, adult_recoverd_stable,
senior_recoverd_stable)

objects = ('child', 'adolescent', 'adults', 'senior')


fig, ax = plt.subplots()

index = np.arange(4)

bar_width = 0.2

opacity = 0.8


rects1 = plt.bar(index - bar_width, deaths_critical, bar_width,

alpha=opacity,

color='maroon',

label='death/critical')


rects2 = plt.bar(index, recovered_stable, bar_width,

alpha=opacity,

color='olivedrab',

label='recovered/stable')


plt.title(' Outcomes vs Age')

plt.xticks(index , objects)

plt.legend()

plt.tight_layout()

plt.show()


male_adolescent = len((deaths_critical.query('age_category_adolescent == "1" and sex ==
"0"')).axes[0])
```

```
male_adult = len((deaths_critical.query('age_category_adult == "1" and sex == "0"')).axes[0])
male_child = len((deaths_critical.query('age_category_child == "1" and sex == "0"')).axes[0])
male_senior =len((deaths_critical.query('age_category_senior == "1" and sex == "0"')).axes[0])
female_adolescent = len((deaths_critical.query('age_category_adolescent == "1" and sex == "1"')).axes[0])
female_adult = len((deaths_critical.query('age_category_adult == "1" and sex == "1"')).axes[0])
female_child = len((deaths_critical.query('age_category_child == "1" and sex == "1"')).axes[0])
female_senior =len((deaths_critical.query('age_category_senior == "1" and sex == "1"')).axes[0])
objects = ('child', 'adolescent', 'adult', 'senior')
male = (male_child, male_adolescent, male_adult, male_senior)
female = (female_child, female_adolescent, female_adult, female_senior)

fig, ax = plt.subplots()
index = np.arange(4)
bar_width = 0.2
opacity = 0.8

rects1 = plt.bar(index - bar_width, male, bar_width,
alpha=opacity,
color='saddlebrown',
label='male')

rects2 = plt.bar(index, female, bar_width,
alpha=opacity,
color='peachpuff',
label='female')


plt.title('Outcome vs gender along with age')
plt.xticks(index , objects)
```

```
plt.legend()


plt.tight_layout()
plt.show()


# # Normalise Feature Set


def normalize(column):
    upper = column.max()
    lower = column.min()
    y = (column - lower)/(upper-lower)
    return y


# # XGBoost Classifier
# With Nan
X = df.drop(['outcome'], axis = 1)
X = X.select_dtypes(include=['int', 'float64', 'uint8'])
y = df['outcome']
x = X.apply(pd.to_numeric)
X = X.applymap(lambda x: np.log(x+1) if x >= 0 else -np.log(-x+1))
X = normalize(X)


import sklearn
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.25,
                                                    shuffle = True,
                                                    random_state=0)
params = {'random_state': [0],
                'eta': [0.02, 0.03, 0.05],
```

```
        'gamma': [2, 4, 6],
        'max_depth': [2, 3, 4],
        'scale_pos_weight': [0.5, 0.9, 1],
        'n_estimators': [200, 300, 400]}

clf = GridSearchCV(xgb.XGBClassifier(), params, cv = 5)
clf.fit(X_train, y_train)
best_grid #Best combination of the parameters
# fit the model with the best grid otained from the gridsearch - the reason why i have duplicated this is so i can plot the feature importance easily. The grid search does not have this option to do it easily (at least from what i could find)
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.25,
                                                    shuffle = True,
                                                    random_state=0)

clf = xgb.XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
      colsample_bytree=1, eta=0.02, gamma=2, learning_rate=0.1,
      max_delta_step=0, max_depth=3, min_child_weight=1, missing=None,
      n_estimators=200, n_jobs=1, nthread=None,
      objective='binary:logistic', random_state=0, reg_alpha=0,
      reg_lambda=1, scale_pos_weight=1, seed=None, silent=True,
      subsample=1)

clf.fit(X_train, y_train)

from xgboost import plot_importance
plot_importance(clf._Booster)
plt.show()
```

```
y_train_pred =    clf.predict(X_train)
y_test_pred  =    clf.predict(X_test)
y_train_pred_pr = clf.predict_proba(X_train)[:,-1]
y_test_pred_pr =  clf.predict_proba(X_test)[:,-1]

auc_score, accuracy, precision, recall, f1 = test_results(y_test, y_test_pred, y_test_pred_pr)



# # KNN Classifier
# Fillna(-1)
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score, GridSearchCV

X = df.drop(['outcome'], axis = 1)
X = X.select_dtypes(include=['int', 'float64', 'uint8'])
y = df['outcome']

X = X.apply(pd.to_numeric)
X = X.applymap(lambda x: np.log(x+1) if x >= 0 else -np.log(-x+1))
X = normalize(X)
X = X.fillna(-1)
import sklearn

X_train, X_test, y_train, y_test = train_test_split(X,
                                    y,
                                    test_size=0.245,
                                    shuffle = True,
                                    random_state=4)
clf = KNeighborsClassifier(n_neighbors = 3)
clf.fit(X_train,y_train)
```

```
y_train_pred =    clf.predict(X_train)
y_test_pred  =    clf.predict(X_test)
y_train_pred_pr = clf.predict_proba(X_train)[:,-1]
y_test_pred_pr =  clf.predict_proba(X_test)[:,-1]
auc_score, accuracy, precision, recall, f1 = test_results(y_test, y_test_pred, y_test_pred_pr)

#Cross validation

#create new a knn model
knn2 = KNeighborsClassifier()

#create a dictionary of all values we want to test for n_neighbors
param_grid = {'n_neighbors': np.arange(1, 25)}

#use gridsearch to test all values for n_neighbors
knn_gscv = GridSearchCV(knn2, param_grid, cv=5)

#fit model to data
knn_gscv.fit(X, y)

print('******************************CROSS
VALIDATION****************************** \n')
#check top performing n_neighbors value
print('Top performing n_neighbors value:', knn_gscv.best_params_)

#check mean score for the top performing value of n_neighbors
print('mean score for the top performing value of n_neighbors:', knn_gscv.best_score_)

# # KNN using fillna() Visualisation
from matplotlib.colors import ListedColormap
```

```
import pylab as pl

# Create color maps for 5-class classification problem
label_class = ['Critical/Died', 'Recovered/Stable']
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA'  ])
cmap_bold = ListedColormap(['#FF0000', '#00FF00'])

X_train = np.array(X_train)
x_min, x_max = X_train[:, 1].min() - .1, X_train[:, 1].max() + .1
y_min, y_max = X_train[:, 4].min() - .1, X_train[:, 4].max() + .1

xx, yy = np.meshgrid(np.linspace(x_min, x_max,19),
                     np.linspace(y_min, y_max,19))

Z= y_test_pred
# Put the result into a color plot
Z = Z.reshape(xx.shape)
pl.figure()
pl.pcolormesh(xx, yy, Z, cmap=cmap_light)

# Plot also the training points
scatter = pl.scatter(X_train[:, 1], X_train[:, 4], c=y_train_pred, cmap=cmap_bold)
# pl.legend(handles=scatter.legend_elements()[0], labels=label_class, frameon=False)
pl.colorbar(label=label_class)
# pl.title('Age and connection to Wuhan effecting health')
pl.xlabel('connection to wuhan')
pl.ylabel('No of chronic diseases')
pl.axis('tight')
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.ticker import LinearLocator, FormatStrFormatter
```

```
fig = plt.figure(1, figsize=(20, 15))
ax = Axes3D(fig, elev=48, azim=134)

# surf= ax.plot_surface(xx, yy, Z, cmap=cmap_light,
#                       linewidth=0, antialiased=False)

# # Customize the z axis.
# ax.set_zlim(-1.01, 1.01)
# ax.zaxis.set_major_locator(LinearLocator(10))
# ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))

# # Add a color bar which maps values to colors.
# fig.colorbar(surf, shrink=0.5, aspect=5)

ax.scatter(X_train[:, 0], X_train[:, 1], X_train[:, 2], c=y_train_pred,
          cmap=cmap_bold, edgecolor='k', s = X_train[:, 3]*50)

for name, label in [('Died/Critical', 0), ('stable/Recovered', 1)]:
    ax.text3D(X_train[y_train_pred == label, 0].mean(),
              X_train[y_train_pred == label, 1].mean(),
              X_train[y_train_pred == label, 2].mean(), name,
              horizontalalignment='center',
              bbox=dict(alpha=.5, edgecolor='w', facecolor='w'),size=25)

ax.set_title("3D KNN", fontsize=40)
ax.set_xlabel("sex ", fontsize=25)
ax.w_xaxis.set_ticklabels([])
ax.set_ylabel("connection to wuhan", fontsize=25)
ax.w_yaxis.set_ticklabels([])
ax.set_zlabel("chronic diseases", fontsize=25)
```

```
ax.w_zaxis.set_ticklabels([])
plt.show()

# # Using KNN Imputer
from fancyimpute import KNN

X = df.drop(['outcome'], axis = 1)
X = X.select_dtypes(include=['int', 'float64', 'uint8'])
y = df['outcome']
X = X.apply(pd.to_numeric)
X = X.applymap(lambda x: np.log(x+1) if x >= 0 else -np.log(-x+1))
X = normalize(X)
df_knn_impute = KNN(k=2).fit_transform(X)


# # KNN using fancy impute
#Fancy Impute

X = df_knn_impute
y = df['outcome']
X_train, X_test, y_train, y_test = train_test_split(X,
                                    y,
                                    test_size=0.245,
                                    shuffle = True,
                                    random_state=4)
clf = KNeighborsClassifier(n_neighbors = 3)
clf.fit(X_train,y_train)
y_train_pred =    clf.predict(X_train)
y_test_pred  =    clf.predict(X_test)
y_train_pred_pr = clf.predict_proba(X_train)[:,-1]
```

```
y_test_pred_pr =  clf.predict_proba(X_test)[:,-1]
auc_score, accuracy, precision, recall, f1 = test_results(y_test, y_test_pred, y_test_pred_pr)

#Cross validation

#create new a knn model
knn2 = KNeighborsClassifier()

#create a dictionary of all values we want to test for n_neighbors
param_grid = {'n_neighbors': np.arange(1, 25)}

#use gridsearch to test all values for n_neighbors
knn_gscv = GridSearchCV(knn2, param_grid, cv=5)

#fit model to data
knn_gscv.fit(X, y)
print('******************************CROSS
VALIDATION****************************** \n')
#check top performing n_neighbors value
print('Top performing n_neighbors value:', knn_gscv.best_params_)

#check mean score for the top performing value of n_neighbors
print('mean score for the top performing value of n_neighbors:', knn_gscv.best_score_)

#KNN using fancy impute Visualization

X_train = np.array(X_train)
x_min, x_max = X_train[:, 1].min() - .1, X_train[:, 1].max() + .1
y_min, y_max = X_train[:, 4].min() - .1, X_train[:, 4].max() + .1
```

```
xx, yy = np.meshgrid(np.linspace(x_min, x_max,19),
                     np.linspace(y_min, y_max,19))

Z= y_test_pred
# Put the result into a color plot
Z = Z.reshape(xx.shape)
pl.figure()
pl.pcolormesh(xx, yy, Z, cmap=cmap_light)

# Plot also the training points
scatter = pl.scatter(X_train[:, 1], X_train[:, 4], c=y_train_pred, cmap=cmap_bold)
# pl.legend(handles=scatter.legend_elements()[0], labels=label_class, frameon=False)
pl.colorbar(label=label_class)
# pl.title('Age and connection to Wuhan effecting health')
pl.xlabel('connection to wuhan')
pl.ylabel('No of chronic diseases')
pl.axis('tight')
```