Covid-19 Data Mining Final Report

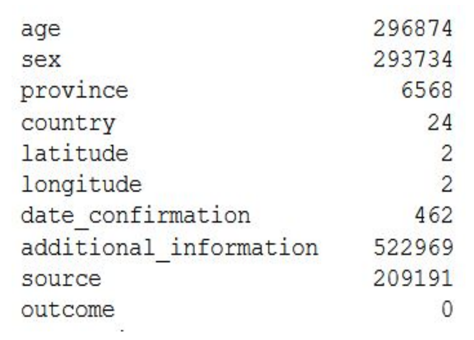
1. Introduction

Covid-19 disease has been spread all over the world in 2020. Millions of people have been diagnosed with covid-19 and many data collection agencies have gathered the covid-19 data through researchers and volunteers. The two data sets that we have used are open source Covid-19 data sets. These datasets contain information of people who have been diagnosed with covid-19 in many different countries all around the world. The information in both data sets has a wide range of attributes related to countries and people. The purpose of our project is to use the information provided in our data sets to predict the final outcome values which are unhospitalize, hospitalized, recovered and deceased. In the first stage, it requires us to explore and visualize the data for each attribute for both datasets and after exploring each feature then we focus on data cleaning, imputing missing values and outliers. After filling up the data we merge both dataset on country and province so that we can use our final dataset in our Random Forest, SVM and XGBoost models to predict the outcome values.

2. Covid-19 Datasets Description and Exploratory Data Analysis

The first data set provides the covid-19 cases by individuals and has length 557,364. The main features from this data set are Age, Sex, Province, Country, Latitude, Longitude, Data Confirmation, Additional Information, Source and Outcome. In this data set Sex, Province, Country, Additional Information and Source are categorial data and the rest are numerical data. The missing values were found in each feature as shown in Figure 2.0.

Age is defined in 362 different patterns which includes patterns like '12-25', '5+', '60-', '16 group', and '11 months'.Sex is defined in 3 different ways 'male':145,583, 'female':118,047 and 'nan':293,734. There are 1,179 unique Provinces where Provinces that have the highest number of data are: Maharashtra 106,515, Lima 48,582, Tamil Nadu 35,939, Delhi 29,173 and Gujarat 28,613. This is approximately 45% of all data. Out of 135 unique Countries, the top 5 Countries with highest data are:India 301144, Peru 95540, Germany 54893, Colombia 19068, Philippines 1717. Containing 87.5% of total data.



```
age                        296874
sex                        293734
province                     6568
country                        24
latitude                        2
longitude                       2
date_confirmation             462
additional_information     522969
source                     209191
outcome                         0
```

Figure 2.0: NAN values by Individual data set

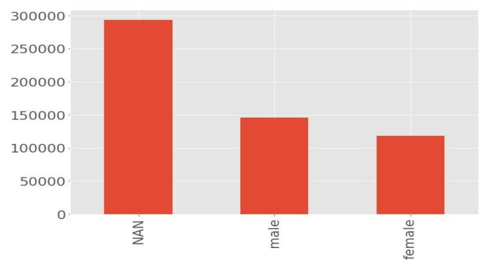


Figure 2.1: Sex Distribution

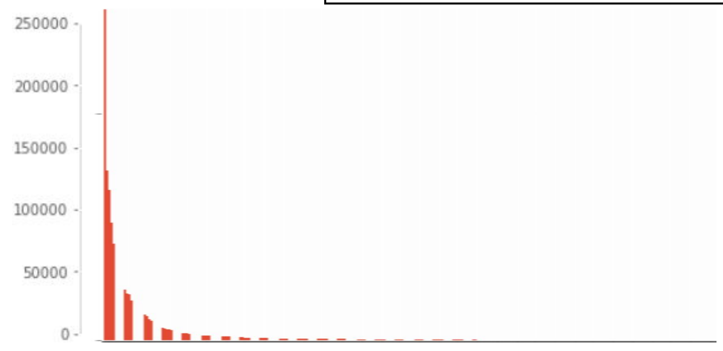


Figure 2.2: Cases distribution by Province

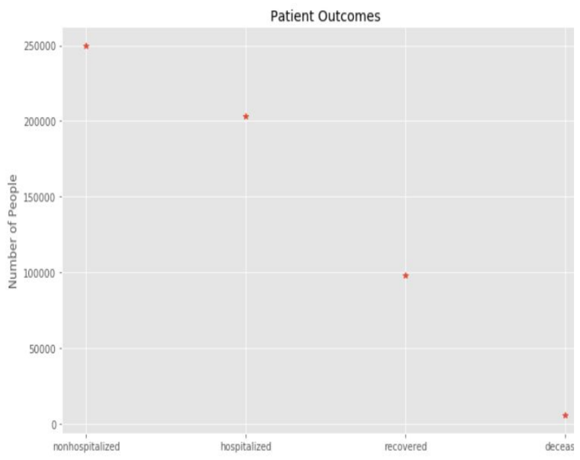


Figure 2.3: Outcome Distribution

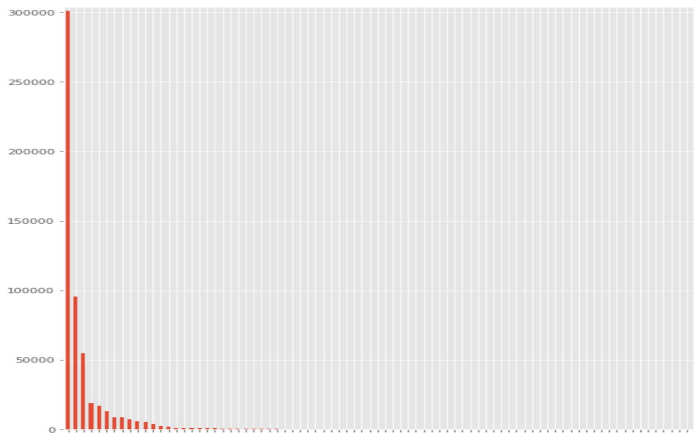
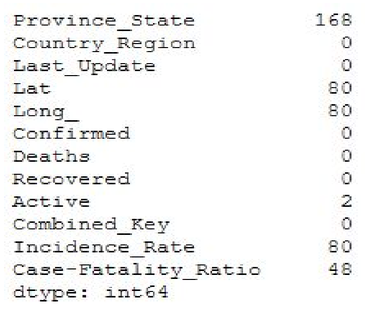


Figure 2.4: Cases Distribution by Country

The latitude and longitude values describe the location of a Province in a Country. The "date_confirmation" is when the Covid-19 case was recorded and its format is day.month.year.. The additional information shows the trace and record of the past history of individuals, who have been diagnosed with Covid-19 but almost 93.8% of data is empty. The "source" attribute contains the hyperlink to the website from where the data is recorded. The "outcome" attribute has 4 unique values which are: 'non-hospitalized', 'hospitalized', 'recovered' and 'deceased'.

The second data provides the Covid-19 cases by Countries. The main features from this data set are: Province, Country, Last Update, Latitude, Longitude, Confirmed, Deaths, Recovered, Active, Combined Key, Incidence Rate and Case Fatality Ratio. In this data set Province, Country and Combined Key are categorical data and remaining features are numerical data types. The missing values found in this data set is shown below in FigureX.



```
Province_State          168
Country_Region            0
Last_Update               0
Lat                      80
Long_                    80
Confirmed                 0
Deaths                    0
Recovered                 0
Active                    2
Combined_Key              0
Incidence_Rate           80
Case-Fatality_Ratio      48
dtype: int64
```

Figure 2.5: NAN values by Country Dataset

Province_State shows the State and Province of each country and has 562 unique values. The Country_Region has each country name which has 188 unique countries. Last_Update is the date and time when records were last updated. Lat and Long_ defines the particular location of the country.

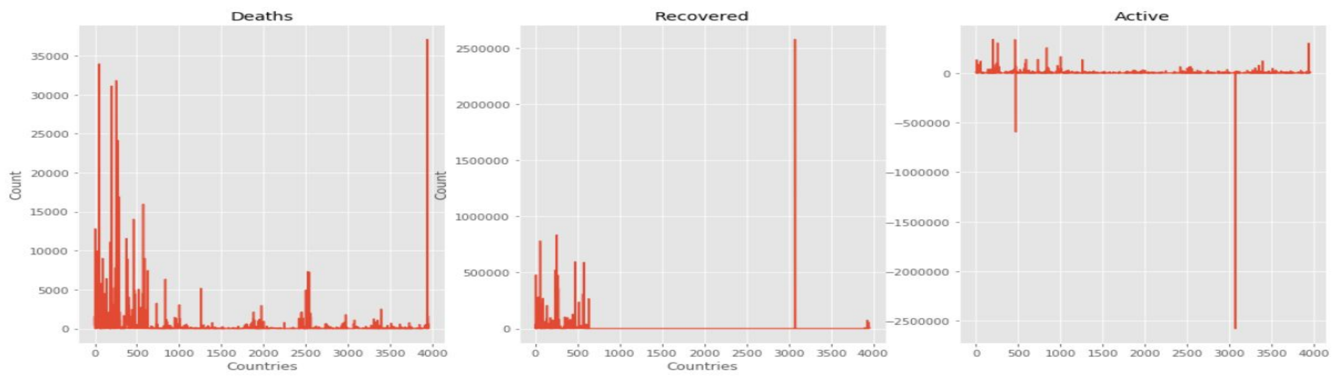


Figure 2.6: Distribution of Deaths, Recovered and Active Cases

Confirmed cases show the number of people that have been diagnosed with Covid-19, Deaths show the total number of deaths Recovered attribute shows the number of people that have been recovered. Active shows the number of active cases. The incidence rate is the number of people who fell ill out of 100,000. Its minimum value is 0 and maximum value is 14871.18. The Case Fatality is the percentage of ratio of total death in a country by Covid-19 and total confirmed cases.

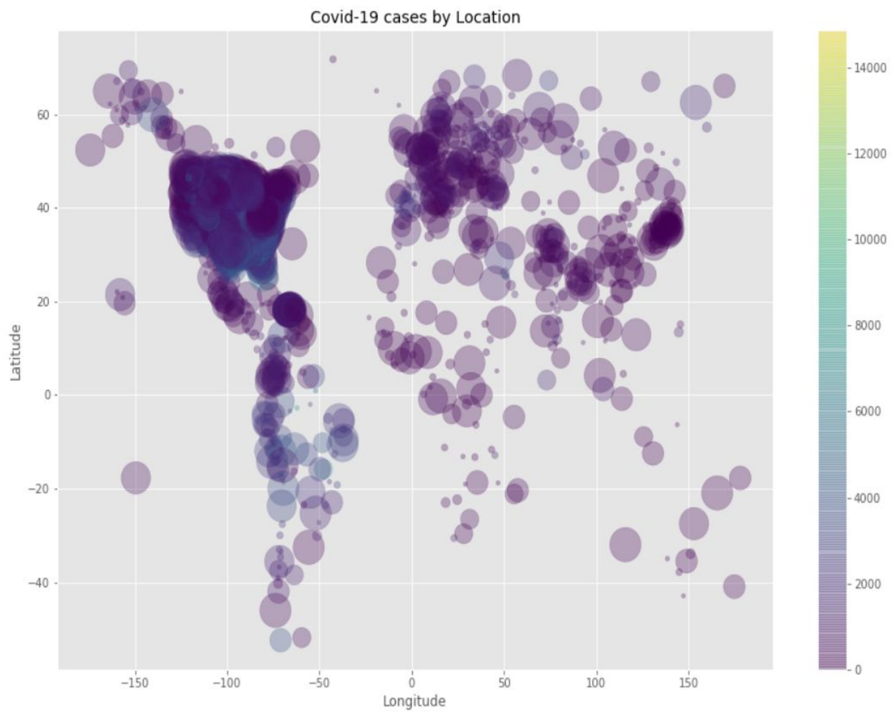


Figure 2.7: Covid Cases by Location

3. Data Preparation

For ages with limits like ‘34-60’, ‘80+’ and ‘8 months’, we used different approaches. If the data is of type‘34-60’, a random number between this limit. For “80+”, a random number between 80 and 95 is used. Data with ‘months’ is converted to (number of months)/12. For the missing age values, we used the mean of age of the same Provinces. For remaining ‘nan’ values, we took a mean by grouping by Country. The figure A shows the histogram plot of how age is distributed after filling the empty data. We followed the same approach for latitude, longitude, Incidence_Rate and Case-Fatality_Ratio.

To fill the 'nan' values in ‘sex’, we find the ratio of male and female for each Province then used this ratio to fill the ‘nan’ values for that Province. Provinces have 6568 ‘nan’ values, most Provinces whose value is ‘nan’ belongs to small countries which do not have Provinces. So we assign the Country value to the Province value. Similarly the ‘Country’ contains 24 ‘nan’ values, where 2 rows are completely ‘nan’ which is considered as wrong data and will be removed while other 22 were given their Province names. ‘date_confirmation’, ‘additional_information’ and ‘source’ values which have ‘nan’ were replaced by ‘Unknown’.

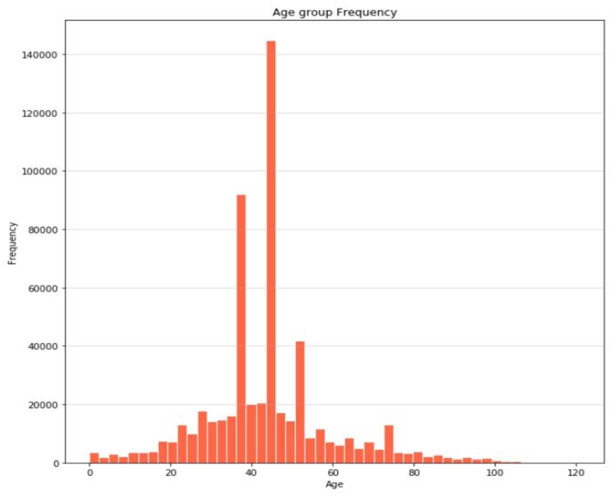


Figure 3.0: Age Group Frequency

For the second data set, we again have Province_State containing 168 ‘nan’ values for the same reason and ‘Country’ values are used again to fill them. Latitude and Longitude contain 80 ‘nan’ values, we use the group by mean for latitude and longitude for each province and then assign the mean average value of that province to fill the 'nan' value. Same strategy is used for 'incidence_rate'. At last for Case fatality ratio we fill the missing value by calculating the ratio of death cases over confirmed cases.

The age has some outliers and we use groups by means for the province and assign the outliers those values. We made sure latitudes are in a range of (-90,90) and longitudes are in the range of (-180,180). There were not many outliers as most data was valid after the cleaning, some of it was already removed which made it easier to deal with outliers. There were some negative values found in "active" cases, which isn't possible so they were removed. We found the fatality rate as '108' which is also not possible, so we removed it.

For merging the two datasets we used Provinces and Countries features. Initially if all Countries in locations dataset are named the same in cases dataset. The location dataset has more unique countries. So we thought our comparison should show that all Countries in locations dataset must be in cases dataset. We found 8 countries that didn't match, by printing them out we identified that 6 countries were named differently in cases dataset and two countries were actually provinces (Puerto Rico and Taiwan). Puerto Rico had cities in Provinces which had coordinates near each other so we replaced it with Puerto Rico and made the Country 'US'. For Taiwan we made it a country and for other different names but same countries such as (Korea, South and South Korea) we gave them a common name. We did the similar comparison for Provinces where cases had 1180 but locations only had 726. Out of which 376 matched. We compared them again by making them all lowerCase and 385 matched this time. So now our data could match most Provinces. We first joined it using Provinces and then the leftover rows were joined using Countries.

The data was categorical and sklearn takes numerical data for machine learning classifiers, so to convert data into numeric values we used get_dummies which converted the categorical column into indicator variables. After converting all the categorical data, we ended up 1401 columns.

4. Classifications Models

We worked with three distinct Classification models, we choose them due to the following reasons:

KNN

K nearest neighbour stores all available cases and classifies new cases based on a similarity measure. This measure can be a distance function such as Euclidean distance and Manhattan distance for a continuous data. In case of categorical data, Hamming distance can be used. In our case we use Minkowski distance.

$$\text{Minkowski distance} \left(\sum_{i=1}^{k} (|x_i - y_i|)^q \right)^{1/q}$$

It makes decisions by taking distances between a query and selecting k examples from the data which are closest to that query, then votes for the most frequent label.

We chose this model because:

- ➔ There is no training period means it does not learn anything in training period. It stores the training data and only learns from it at the time of making real time predictions

- ➔ It is a non-parametric algorithm, which is why there are no assumptions to be made to implement KNN
- ➔ New data can be added as there is no training that takes place before making predictions

Random Forest Classifier
It uses a set of decision trees from a randomly selected subset of the dataset. It then aggregates the votes from different decision trees to decide the final class of the test object. The hyperparameters used are the number of trees in the forest and depth. Gini index or entropy is used for information gain. We chose this model because:

- ➔ It is one of the most accurate learning algorithms
- ➔ It is less impacted by noise
- ➔ No feature scaling is required as it uses a rule based approach instead of distance calculation.

XGBoost (XGBClassifier)
It is a decision-tree based Machine learning algorithm that uses a gradient boosting framework. This classifier is known for good performance, due to several reasons:

- ➔ It approaches the process of sequential tree building using parallelized implementation
- ➔ Tree Pruning: The stopping criterion for tree splitting within GBM framework is greedy in nature and depends on the negative loss criterion at the point of split
- ➔ Hardware Optimization: The algorithm has been designed to make efficient use of hardware resources

5. Initial evaluation

Initially, an 80:20 split of data was used to train and test the models. In the baseline models, Gaussian Naive Bayes was used, but we decided to replace it with K-nearest neighbour. The reason being the low accuracy of GaussianNB and possibility of hyperparameter tuning on K-nearest neighbours.

We imported the saved models and used them to predict the test data, also we calculated the k-fold cross-validation score for each model. K-fold accuracy is a good metric because it's a way to resample the data on k distinct folds i.e k distinct train and test splits, this ensures that every observation from the original dataset has the chance of appearing in the training and test set and would reduce any bias in the sampling process.

The evaluation process involves the accuracy score of test and train data. The confusion matrix was implemented for each machine learning model. Through the matrix, we were able to find the true positive, true negative, false positive and false negative values for each class.

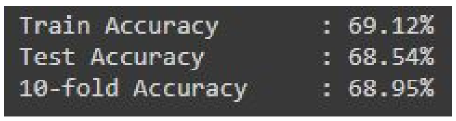
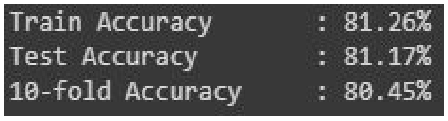
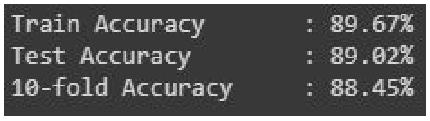

```
Train Accuracy       : 69.12%
Test Accuracy        : 68.54%
10-fold Accuracy     : 68.95%
```

```
Train Accuracy       : 81.26%
Test Accuracy        : 81.17%
10-fold Accuracy     : 80.45%
```

```
Train Accuracy       : 89.67%
Test Accuracy        : 89.02%
10-fold Accuracy     : 88.45%
```

Fig 5.1: (a) Gaussian Model Score (b)Random-Forest Model Score (c) XGboost Score

The most important things to note here were the training score, testing score, and k-fold accuracy. If the training score is very less than the testing score, this means that our model is overfitting on the training data and not performing well on the test data (i.e. not predicting well on testing data / unseen data). The K-fold accuracy was used to reduce the performance bias on one set of train-test splits. It was noticed that the baseline models were performing good on test data, hence not overfitting.

6. Hyperparameter tuning and Results

There were many possible approaches for hyperparameter tuning. We used GridsearchCV in our approach. For the specified parameters of our chosen classifiers, an exhaustive search was done to find the best hyperparameters.

1. **KNN:** We used two hyperparameters, weight and n_neighbors. For weight, 'uniform' and 'distance', and for n-neighbors we had n_neighbors=[2, 3, 5, 7, 9, 11, 13, 20].
2. **Random Forest:** We used two hyperparameters, max_depth and n_estimators. We had our max_depth=[None, 15, 25, 35, 50] and n_estimators=[50, 75, 100, 150, 200].
3. **XGBoost:** We used 2 estimators, max_depth and min_child_weight. We had our max_depth=[10, 20, 30], and min_child_weight=[1 ,3, 5].

The purpose was to maximize the recall for the label "deceased". For this we used the following approach. We did a 75:25 split on our data. We found the recall for the label "deceased" in our training using our classifiers. The max value was 0.08, using the random forest classifier. This was very small. We found that in 557,340 rows, only about 5,999 are with the label "deceased", i.e. only ~1% .

We realized that the purpose of decision making is to find if a patient will end up "deceased", which makes it an important label. Therefore, predicting it accurately is important. Therefore to improve our quality of discovered patterns, we used a technique that we learned in our lecture. We used an advanced sampling method of Biased sampling. In this method, we oversample the minority records. We sampled out 60% of data from the "non-deceased" classes data and 90% of the data from the "deceased" class. In this approach we use a random sample of 330,806 records without the label "deceased " and add 5,398 records with label "deceased" to it ~2%. This gave us better recall results for the training data, for the label "deceased" and slightly better test scores.

For scoring parameters of gridsearchCV, we used accuracy, overall recall, recall on class "deceased". 5-fold accuracy of each model was taken i.e. five distinct splits of train and test were used to get the scores. This 5-fold reduced the performance bias. The results of each model are in results.pdf.

Note: The Train test scores in the graphs are from gridsearchCV's default 90:10 split.

The two best parameters for **KNN** are:

KNN						
Hyperparameters	**Accuracy**		**Overall Recall**		**Recall on 'deceased'**	
Weight, n_neighbour	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
Uniform, 3	**0.865455**	**0.82262**	**0.671029934**	**0.610557135**	**0.160166544**	**0.079386856**
Distance, 20	0.910915	0.84126	0.758641198	0.620190313	0.371345178	0.0765994

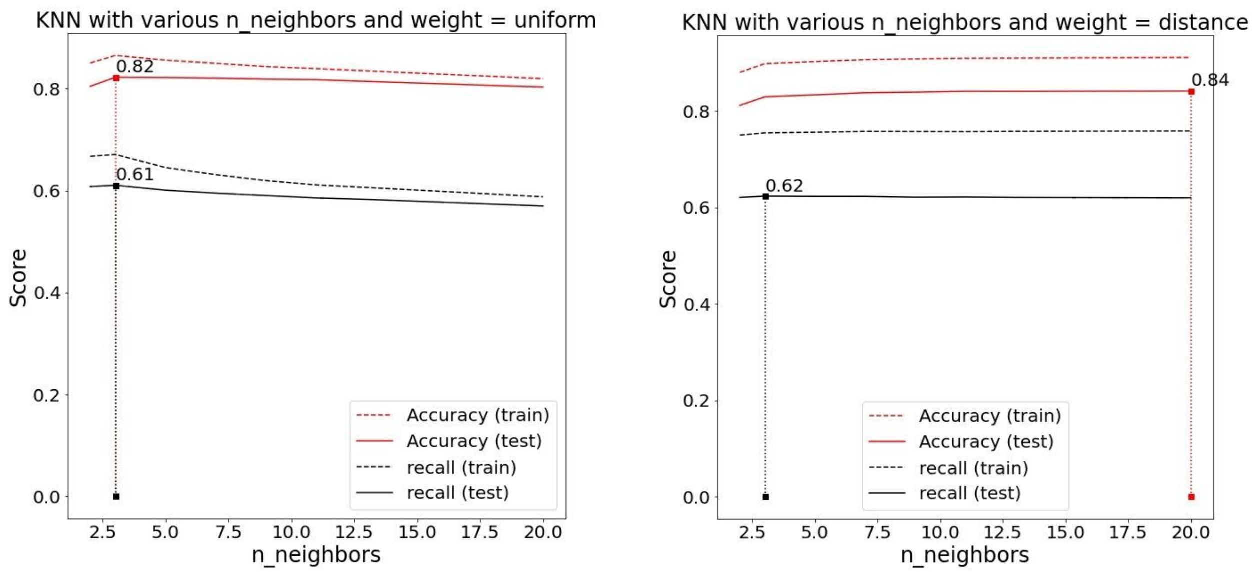
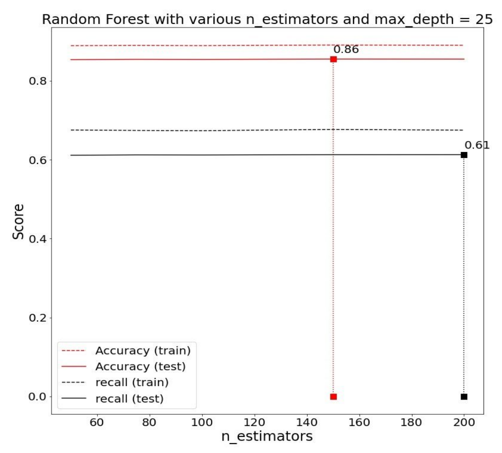


Figure 6.0: KNN Model Score Plot

Out of these two, we decided "Uniform, 3" to be our best parameter. This is because it avoids overfitting, though it gives a little less value on "deceased" but overfitting factor of "Distance, 20" overtakes due to similar total recall

The three best parameters for **Random forest classifier** are:

Random Forest						
Hyperparameters	**Accuracy**		**Overall Recall**		**Recall on 'deceased'**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**
n_estimators=150,max_depth=None	**0.909515**	**0.85336**	**0.76127**	**0.638495**	**0.403151**	**0.112277826**
n_estimators=150, max_depth=35	**0.908475**	**0.85586**	**0.747555123**	**0.633867428**	**0.361948541**	**0.099797239**
n_estimators=100, max_depth=50	**0.909485**	**0.85368**	**0.759997483**	**0.636878877**	**0.39691475**	**0.103735979**



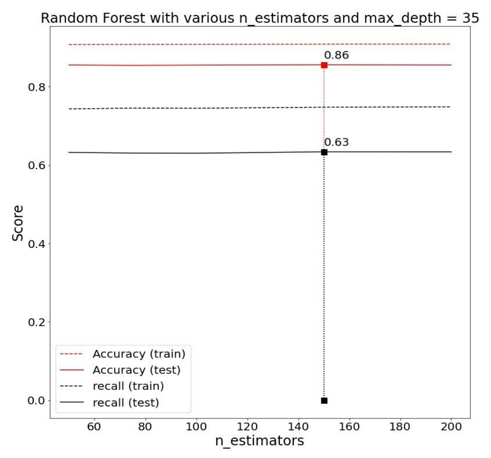
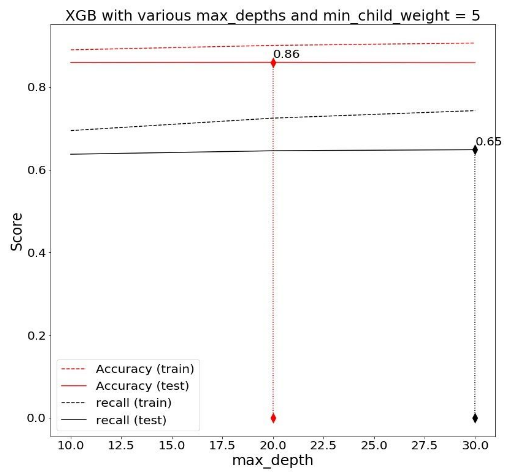


Figure 6.1: Random Forest Score Plot

We didn't choose max_depth=50, None because of over fitting. Max-depth =15 had low test accuracy and recall on "deceased". Out of max_depth=25, 35. We chose max_depth=35 because of it's higher trained overall recall and recall on "deceased".
The four best parameters for **XGB** are:

XGB						
Hyperparameters {min_child_weight, max_depth}	**Accuracy**		**Overall Recall**		**Recall on 'deceased'**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**
1, 20	0.90907	0.85592	0.752999023	0.639753901	0.371438033	0.113945094
3, 20	0.90539	0.85928	0.7404085	0.64817475	0.34105221	0.144163763
3, 30	**0.90922**	**0.8579**	**0.754586991**	**0.648497235**	**0.380572866**	**0.144166183**
5, 30	0.90625	0.85848	0.742711099	0.648311824	0.342444271	0.141388



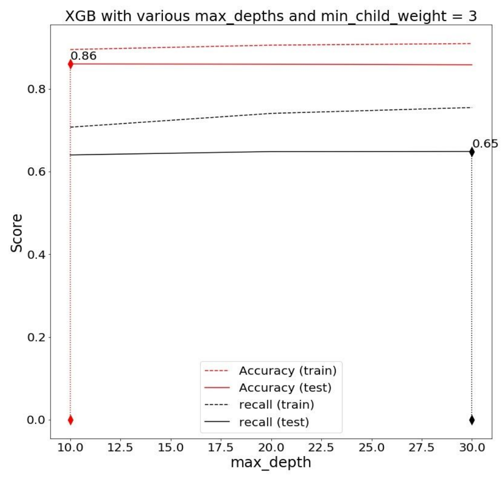
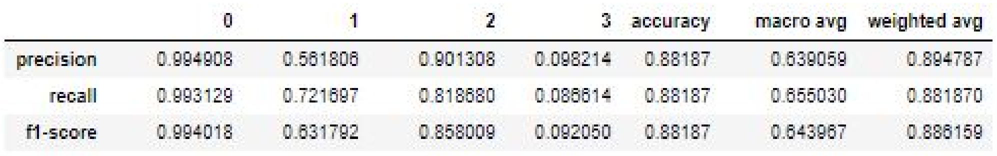


Figure 6.2: XGB Score Plot

We chose min_child_weight as 3 with depth 30. This is because it had a better value for overall recall and recall on ”deceased”, whereas the accuracy was similar for other hyperparameters.

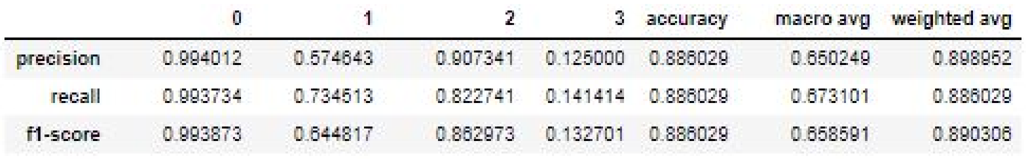
7. Results

For the best parameters of each classifier, we trained and tested on the initial split of train and test and found these results.



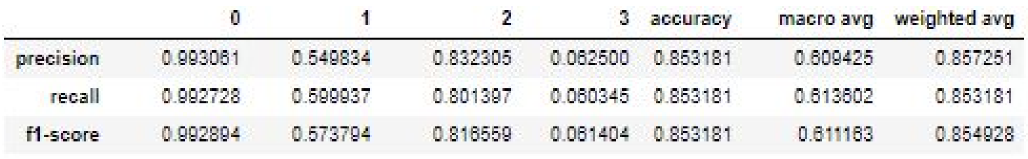
	0	1	2	3	accuracy	macro avg	weighted avg
precision	0.994908	0.561806	0.901308	0.098214	0.88187	0.639059	0.894787
recall	0.993129	0.721697	0.818680	0.086614	0.88187	0.655030	0.881870
f1-score	0.994018	0.631792	0.858009	0.092050	0.88187	0.643967	0.886159

Fig 7.1: Classification report RandomForest Classifier



	0	1	2	3	accuracy	macro avg	weighted avg
precision	0.994012	0.574643	0.907341	0.125000	0.886029	0.650249	0.898952
recall	0.993734	0.734513	0.822741	0.141414	0.886029	0.673101	0.886029
f1-score	0.993873	0.644817	0.862973	0.132701	0.886029	0.658591	0.890306

Fig 7.2: Classification report XG-Boost Classifier



	0	1	2	3	accuracy	macro avg	weighted avg
precision	0.993061	0.549834	0.832305	0.062500	0.853181	0.609425	0.857251
recall	0.992728	0.599937	0.801397	0.060345	0.853181	0.613602	0.853181
f1-score	0.992894	0.573794	0.816559	0.061404	0.853181	0.611163	0.854928

Fig 7.3: Classification report KNN Classifier

8. Conclusion

After running our best selected hyper parameters on test data for all three models. We got the above values as shown in figures 7. The label “3” represents our “deceased”. The accuracy for KNN classifiers is less than the other two classifiers, as well as its recall and precision. It also took a lot of time to train as there is no specific model and it makes real time predictions. Even though the maximum accuracy of XG Boost is similar to that of Random forest classifier. We see that XG-Boost classifier has a better precision as well as recall on “deceased”. The macro average also comes out to be better for XG-Boost. Not only this, the difference between train and test score is less for XG Boost classifier. With this we conclude that **XG-Boost**, with the hyperparameters: **min_child_support-3 , max-depth-30** provides us with the best result.

9. Lessons Learnt and Future Work

The project is a valuable experience for the team to encounter a real-world data science problem. The project followed a great pipeline, starting from data cleaning, filling out missing and redundant data, removing the outliers, joining data from different sources, and then applying various machine learning techniques. Referencing to the course material and performing external research to understand certain concepts was highly beneficial. Furthermore, the training and testing process of machine learning models were practiced and better understood.

In addition, improvement of time management, communication and team working skills are some of the other outcomes. Regular weekly/daily scrum meetings enables each team member to have a meticulous mindset and a well-structured and concise workflow for completion of the project. Most importantly, we experienced working with real data and developed a data mining software to extract information from it. It can be improved further by finding the redundant data and removing it. We found a lot of cases which were added that consisted of the same information but were gathered from different sources. This made recall for the label "deceased" very low, hence taking out the fruitful information.

A milestone should have consisted of removing the redundant data, some reports by the same province were added on overlapping dates. This would have been a very hard task to look in detail.
The biggest problem was that it was hard to run such a big amount of data.

10. Contributions

Harnoor: In Milestone 1, Work on exploratory data analysis and plot the graphs for each feature. Visualize each feature and find all the 'nan' values for each feature. Also work on data cleaning of Dates and Ages features on covid-19 individual dataset. Visualize different kinds of date and age patterns. In Milestone 2, trained and tested Random Forest and implemented a confusion matrix.

Amritpal: In Milestone1, Work on data cleaning of province, country, sex, additional information and source features on covid-19 individual dataset. Deal with outliers in covid-19 by country dataset. Transform and merge the two datasets. In Milestone 2, trained and tested XG Boost model and implemented the functions to calculate accuracy and confusion matrix.

Abhay Jolly: In milestone 1, Work on data cleaning of Province_State, Country_Region, Lat, Long,Incidence rate and case fatality ratio of covid-19 dataset by countries. Deal with the outliers in covid-19 individual data sets.Transform and merge both data sets for final dataset. In Milestone 2, trained and tested GaussianNB and implemented a confusion matrix.

The final milestone was more of pair programming, where we used zoom and worked together on completing the code and running gridSearch and best models on Google Collaboratory.

11. References

1. https://builtin.com/data-science/random-forest-algorithm
2. https://medium.com/@anuuz.soni/advantages-and-disadvantages-of-knn-ee06599b9336
3. https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d
4. https://scikit-learn.org/stable/auto_examples/model_selection/plot_multi_metric_evaluation.html