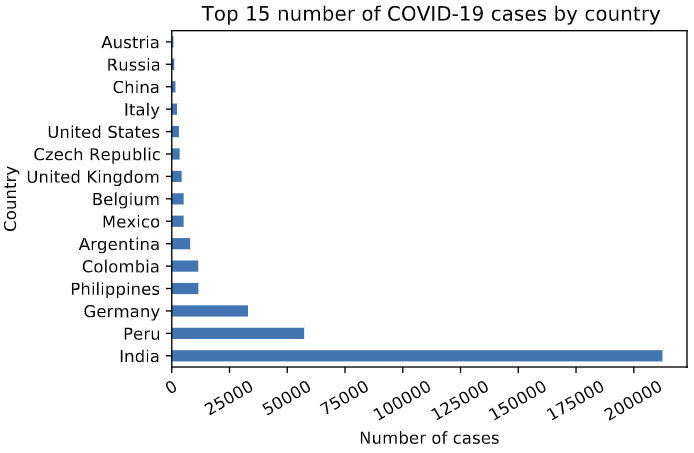
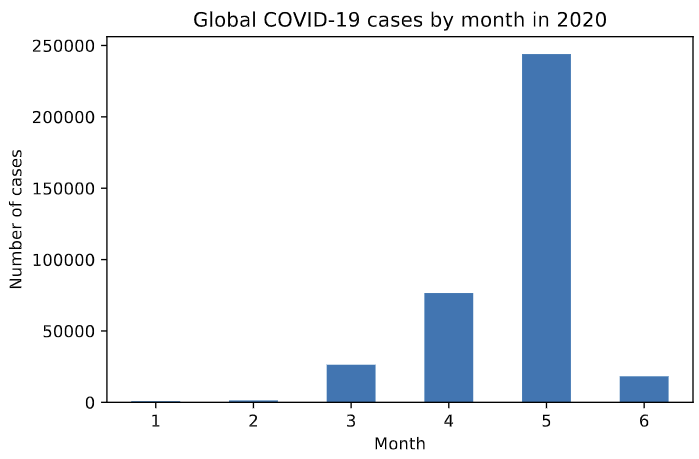
# Problem Statement

The goal of this project was to get first-hand experience working with data. We looked at data collected from COVID-19 cases around the world and were tasked to clean this data and then use classification models on our training data to predict the outcome of a patient from the test data.

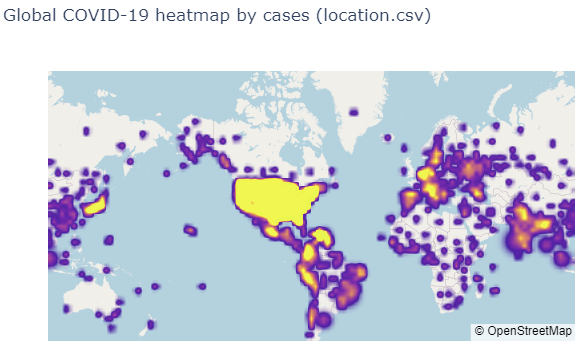
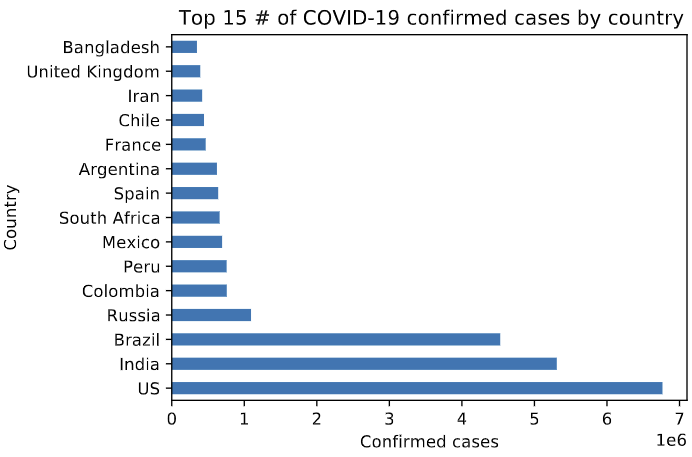
# Dataset Description and EDA

When analyzing the dataset using visualizations, we found the following plots to be the most insightful.

The following two plots were generated using data from cases\_train.csv. These two plots tell us that there were a lot of individual cases collected from India between the months of March and June, but most heavily in May.



To contrast the data above, the next plots were generated using data from location.csv, which contained the number of cases for each location instead of individual cases. We observed many confirmed cases in United States which surprised us considering how few cases there were in cases\_train.csv. This difference tells us that either there were many inaccuracies in data collection, or that either dataset was sampled by convenience since a simple random sample of global COVID-19 statistics should provide similar results.



# Data Preparation

When cleaning data, we looked to replace blank rows with “Unknown” values, and manually cleaned the noisiest column of ages by looking through all unique values and replacing age ranges and dates with valid ages. We also took a manual approach when looking for initial outliers, looking through unique values of all the columns for strange values. We found that rows of data taken from the province of Lima had seemingly erroneous data, with almost all their ages being under 1 or over 100, so we decided to remove those rows.

While transforming data from the location dataset, we aggregated the data of individual counties into their respective states for the US, and aggregated individual provinces into their respective countries for the rest of the world. We then joined this data based on province and country with the training and test data, and only included columns that we felt would give us meaningful knowledge. This included total incidence rate, provincial incidence percentage, total case-fatality ratio, and provincial case-fatality ratio.

# Classification Models

The three classification models that we chose to use were XGBoost, K-Nearest Neighbours, and Random Forests. As an overview, we chose two ensemble methods: XGBoost and Random Forests, because they are robust to outliers and non-linear data, and they produce results with high classification accuracy. As for KNN, we chose it because we felt the results were very simple to interpret and it is robust regarding search space since classes do not have to be linearly separable like in SVM.

First let us discuss our decision to choose XGBoost as our boosting tree variant versus AdaBoost. XGBoost is much faster than AdaBoost and has a multitude of hyperparameters that can be tuned to increase performance, whereas AdaBoost only supports tuning for max depth, learning rate, and number of iterations [1]. XGBoost also has a regularization parameter that reduces variance which AdaBoost lacks.

Next, we will compare Random Forests versus Decision Trees. The problem with Decision Trees is that single decision trees are prone to overfitting (high variance), especially when a tree is grown deep. The classical way to combat this issue is to set a max depth but that increases the error due to bias [2]. Random Forests essentially minimizes the error due to variance and bias by using a collection of decision trees whose results are aggregated into a single result.

Finally, we will compare KNN to SVM. As previously mentioned, we mainly chose KNN since its results are easily interpretable. However, it is important to note that SVM handles outliers better than KNN does [3]. Although in terms of performance, KNN outperforms SVM when the number of training data objects if far greater than the number of features, which in our case is true.

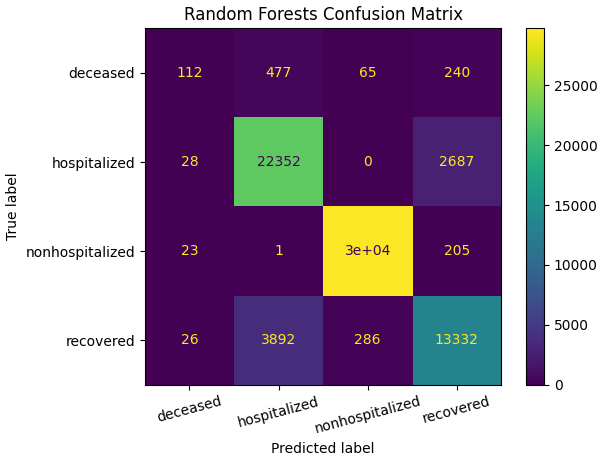
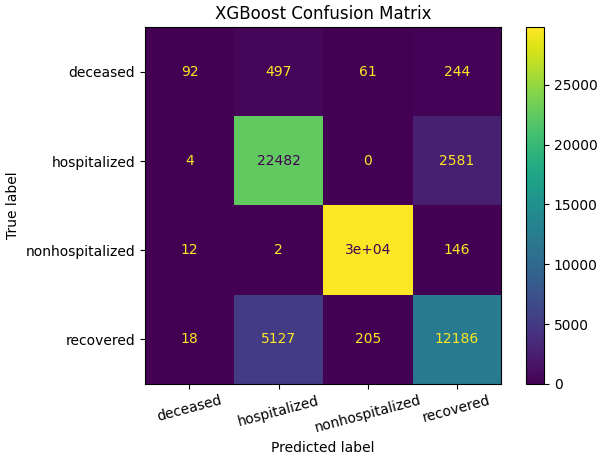
# Initial Evaluation and Overfitting

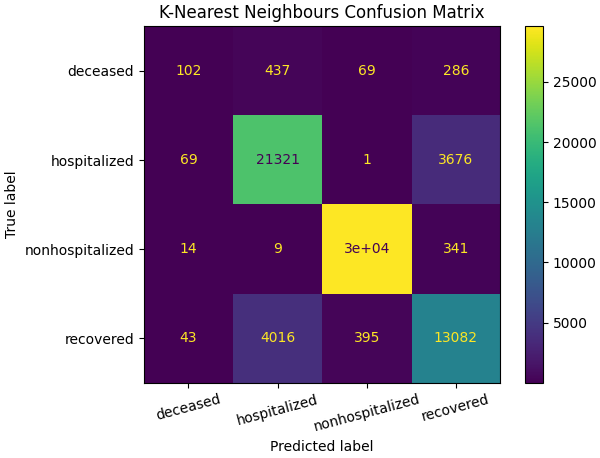
Initially, we considered prediction accuracy as our main metric for evaluating the three classification models. However, we realized that prediction accuracy was not the best metric to use since the COVID-19 dataset was highly imbalanced and therefore, we would have gotten incorrect insights. We decided to use the recall on the ‘Deceased’ outcome label as our main metric. Using scikit-learn’s classification report, we got the recall on the ‘Deceased’ outcome label for each of our models on both the training and validation datasets.

|  |  |  |  |
| --- | --- | --- | --- |
| **Training Dataset** | ‘Deceased’ Recall | ‘Deceased’  Precision | ‘Deceased’  F1-Score |
| XGBoost | 0.1781 | 0.9386 | 0.2994 |
| Random Forests | 0.3393 | 0.9730 | 0.5031 |
| K-Nearest Neighbours | 0.3628 | 0.8679 | 0.5117 |

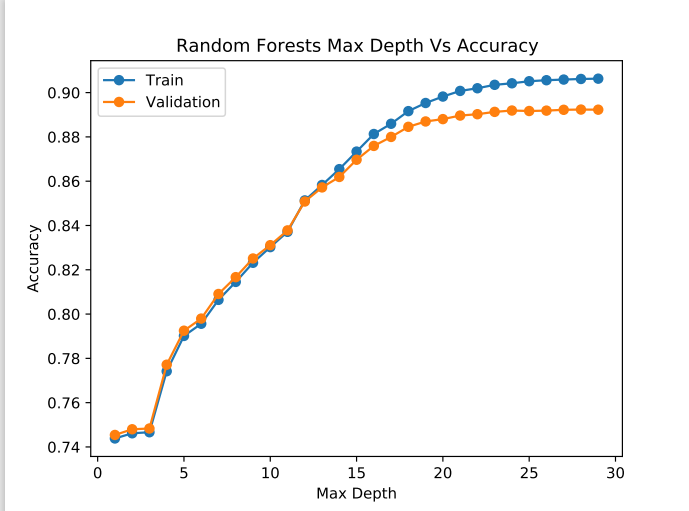
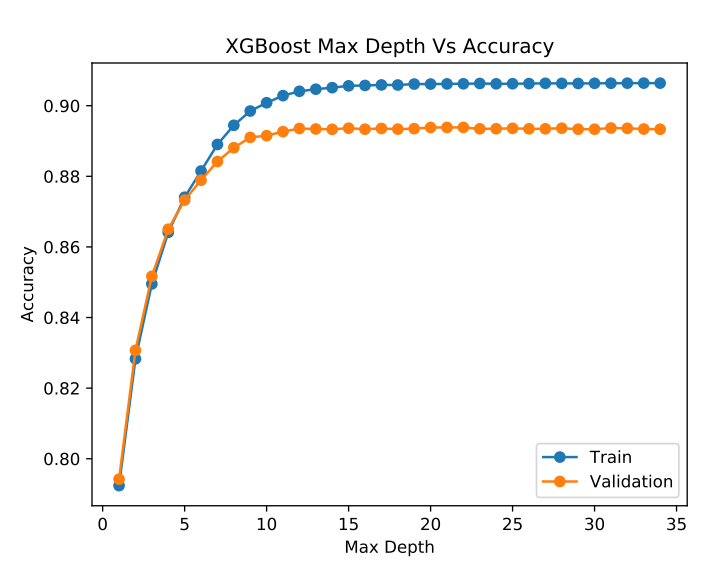
|  |  |  |  |
| --- | --- | --- | --- |
| **Validation Dataset** | ‘Deceased’ Recall | ‘Deceased’  Precision | ‘Deceased’  F1-Score |
| XGBoost | 0.1029 | 0.7302 | 0.1804 |
| Random Forests | 0.1253 | 0.5926 | 0.2068 |
| K-Nearest Neighbours | 0.1141 | 0.4474 | 0.1818 |

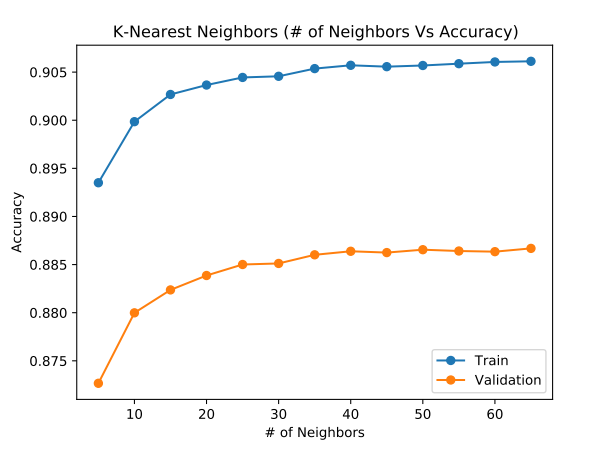
We also used scikit-learn’s confusion matrix on the validation dataset for each of the models. For each confusion matrix C below, we define Cij to be the number of data objects known to be in group *i* (true label) but were predicted to be in group *j* (predicted label). From these results we can derive insights like how each model is incorrectly predicting data objects that are deceased to be hospitalized. This implies that there are more false negatives than true positives for data objects with ‘deceased’ outcome label.





For the classification models we trained, we did not observe any overfitting. We noticed that the accuracy tended to stagnate for both the training and validation data after further increase of the hyperparameters, as can be seen by the plots below. We tested different values for each model’s respective hyperparameter to find the most optimal values for them and take precautions towards overfitting.





# Hyperparameter Tuning

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **XGBoost Hyperparameters** | F1-Score on ‘deceased’ | Recall on ‘deceased’ | Overall Accuracy | Overall Weighted Recall |
| max\_depth = 8,  learning\_rate = 0.4,  n\_estimators = 100 | 0.7751 | 0.7407 | 0.8896 | 0.8896 |
| max\_depth = 18,  learning\_rate = 0.5,  n\_estimators = 100 | 0.7886 | 0.7665 | 0.8939 | 0.8939 |
| max\_depth = 18,  learning\_rate = 0.5,  n\_estimators = 90 | 0.7882 | 0.7647 | 0.8939 | 0.8939 |
| max\_depth = 25,  learning\_rate = 1.0,  n\_estimators = 150 | 0.7872 | 0.7647 | 0.8933 | 0.8933 |
| max\_depth = 18,  learning\_rate = 0.6,  n\_estimators = 100 | 0.7879 | 0.7649 | 0.8936 | 0.8936 |
| max\_depth = 18,  learning\_rate = 0.5,  n\_estimators = 100 | 0.7884 | 0.7660 | 0.8938 | 0.8938 |
| max\_depth = 18,  learning\_rate = 0.5,  n\_estimators = 110 | 0.7885 | 0.7661 | 0.8938 | 0.8938 |
| max\_depth = 18,  learning\_rate = 0.5,  n\_estimators = 80 | 0.7881 | 0.7646 | 0.8939 | 0.8939 |
| max\_depth = 18,  learning\_rate = 0.5,  n\_estimators = 95 | 0.7883 | 0.7655 | 0.8939 | 0.8939 |
| max\_depth = 18,  learning\_rate = 0.5,  n\_estimators = 105 | 0.7884 | 0.7659 | 0.8938 | 0.8938 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Random Forests Hyperparameters** | F1-Score on ‘deceased’ | Recall on ‘deceased’ | Overall Accuracy | Overall  Weighted Recall |
| max\_features = 0.9,  min\_samples\_leaf = 1  n\_estimators = 25 | 0.7852 | 0.7619 | 0.8923 | 0.8923 |
| max\_features = 0.8,  min\_samples\_leaf = 1  n\_estimators = 25 | 0.7854 | 0.7615 | 0.8927 | 0.8927 |
| max\_depth = 23,  min\_samples\_leaf = 1  n\_estimators = 25 | 0.7777 | 0.7478 | 0.8900 | 0.8900 |
| max\_depth = 28,  min\_samples\_leaf = 1  n\_estimators = 25 | 0.7837 | 0.7585 | 0.8920 | 0.8920 |
| max\_depth = 28,  min\_samples\_leaf = 1  n\_estimators = 100 | 0.7842 | 0.7595 | 0.8923 | 0.8923 |
| max\_features = 0.9  max\_depth = 28,  min\_samples\_leaf = 1  n\_estimators = 100 | 0.7865 | 0.7648 | 0.8928 | 0.8928 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **KNN Hyperparameters** | F1-Score on ‘deceased’ | Recall on ‘deceased’ | Overall Accuracy | Overall  Weighted Recall |
| n\_neighbors = 65,  p = 1 (Manhattan),  weights = ‘distance’ | 0.7745 | 0.7526 | 0.8870 | 0.8870 |
| n\_neighbors = 65,  p = 2 (Euclidean),  weights = ‘distance’ | 0.7741 | 0.7514 | 0.8867 | 0.8867 |
| n\_neighbors = 69,  p = 1 (Manhattan),  weights = ‘distance’ | 0.7748 | 0.7531 | 0.8871 | 0.8871 |
| n\_neighbors = 69,  p = 2 (Euclidean),  weights = ‘distance’ | 0.7744 | 0.7521 | 0.8868 | 0.8868 |
| n\_neighbors = 80,  p = 1 (Euclidean),  weights = ‘distance’ | 0.7752 | 0.7534 | 0.8872 | 0.8872 |

# Results

# Conclusion

# Prediction on Test Dataset

# Lessons Learnt and future work

# References

[1] The Ultimate Guide to AdaBoost, random forests and XGBoost [online]

Available: https://towardsdatascience.com/the-ultimate-guide-to-adaboost-random-forests-and-xgboost-7f9327061c4f

[2] Decision Trees and Random Forests [online]

Available: https://towardsdatascience.com/decision-trees-and-random-forests-df0c3123f991

[3] Comparative study on the classic machine learning algorithms [online]

Available: https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222

# Contributions

## Alvin Ho

## Kenrick Lam

Milestone 1: exploratory data analysis, handling outliers, outcome labels

Milestone 2: Random Forests, accuracy and classification report functions, report

Milestone 3:

## Steven Tran

Milestone 1: imputing data, cleaning ages, transforming and joining data

Milestone 2: XGBoost, accuracy and classification report functions, report

Milestone 3: