CMPT459 - COVID19 Outcome Prediction Model
Amosl - 301315811 / Timothy 301307082 / Bianca B 301304375

1. Problem Statement

Two public datasets on Covid Data are provided and the goal of this project is to analyze the possible characteristics of Covid patients. Then we will train models to capture the features that represent the possible outcomes (Recovered, Hospitalized, Nonhospitalized, and Deceased) of these patients. At the end, the models should predict the outcome of an individual correctly if it is provided with the features that the models were trained on. In this part of the project, we specifically focus on the f1-score on deceased as it is deemed most important.

2. Dataset description and EDA

The Covid-19 datasets contain three separate files. The datasets have been frozen on September 20th, 2020. Case dataset for training (cases_train.csv) contains the training data and outcome for individual cases. Case dataset for testing (cases_test.csv) is similar to training data but without the 'outcome' label. Location dataset (location.csv) contains the number of cases and health status for each location.

Table 1 - Summary Table of "cases_train.csv"

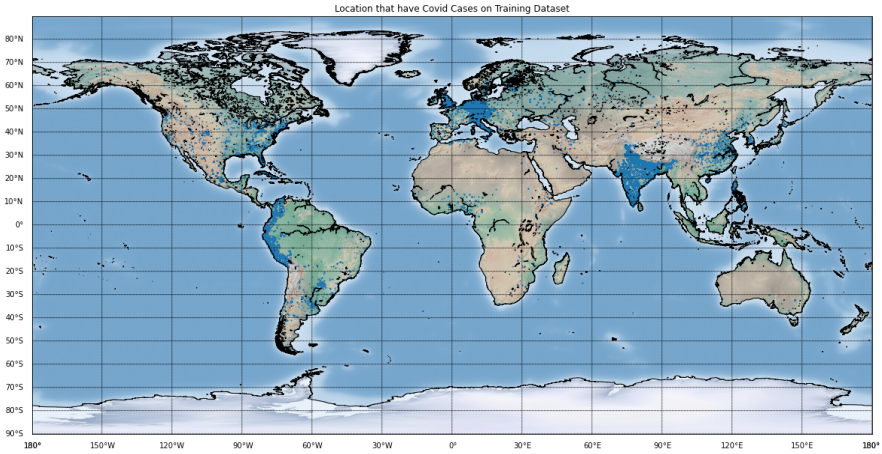
Attributes	Data Type	Number of Missing Values
age	"char"	209265
sex	"char"	207084
province	"char"	4106
country	"char"	18
latitude	"numeric"	2
longitude	"numeric"	2
date_confirmation	"char"	374
additional_information	"char"	344912
source	"char"	128478
outcome	"char"	0

Table 2 - Summary Table of "location.csv"

Attributes	Data Type	Number of Missing Values
Province_State	"char"	168
Country_Region	"char"	0
Last_Update	"char"	0
Lat	"numeric"	80
Long	"numeric"	80
Confirmed	"int"	0
Deaths	"int"	0
Recovered	"int"	0
Active	"numeric"	2
Combined_Key	"char"	0
Incidence_Rate	"numeric"	80
Case-Fatality_Ratio	"numeric"	48

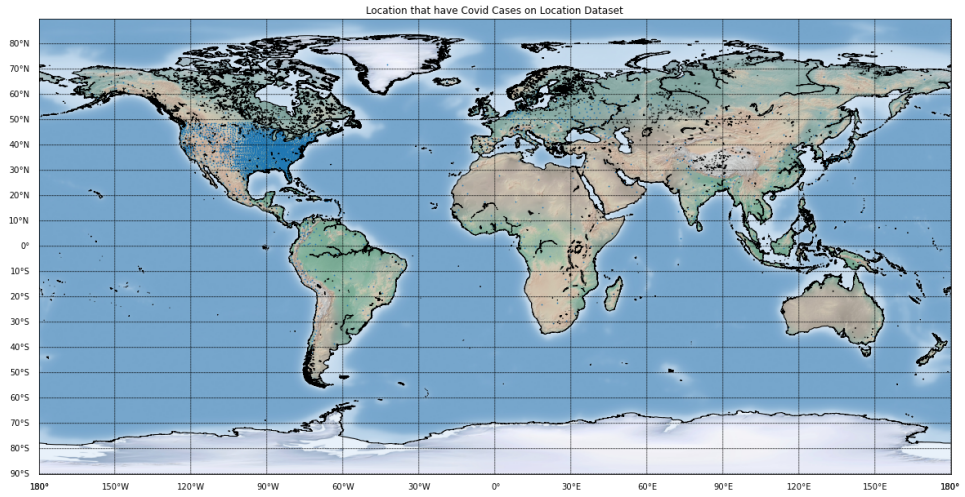
We created various visualizations to model the attribute distribution based on the value and type of data. For example, we plotted geographic data (the combination of longitude and latitude) on a world map to visualize the world-wide distribution of Covid cases contained in our dataset:

Graph 1- Case distribution from "cases_train.csv" based on latitude and longitude



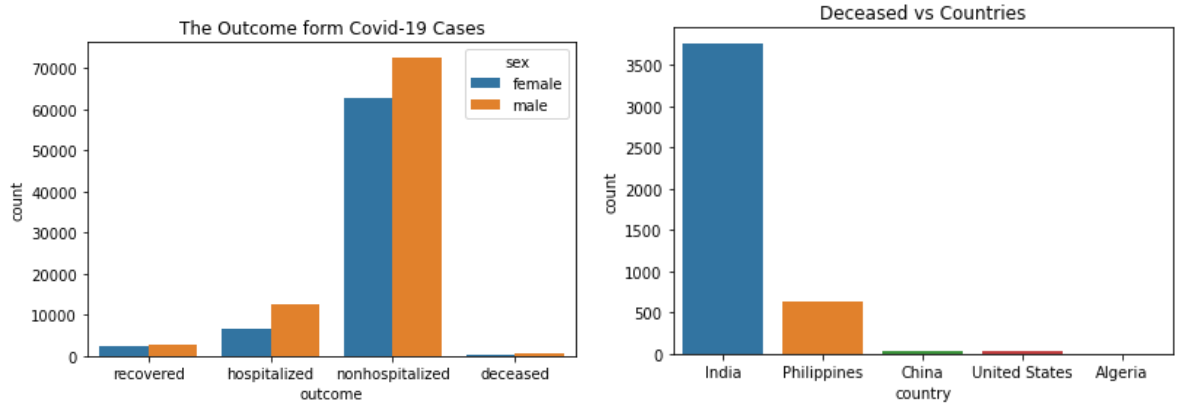
This map helped demonstrate where the majority of our dataset is located, such as the clusters in India and Europe. It also shows that our data is clustered, opposed to being evenly distributed, and that regions such as Africa and Oceania have very little representation. It also helped identify possible outliers, as it seems there are some cases in the ocean.

Graph 2- Case distribution from “location.csv” based on latitude and longitude



In the location dataset, most of the cases are located in the US. Further inspection of the dataset reveals this to be caused by the US listing each county as a unique row, as opposed to each state under the province label.

Graph 3 - Histogram on Sex and Outcome / Deceased vs Countries from “cases_train.csv”



Most of the covid cases are non-hospitalized, and the vast majority of our dataset does not contain cases that resulted in death. It is interesting to note the relative gender parity in recovered and deceased however, when compared to our data for hospitalized and nonhospitalized.

3. Data preparation

3.1 Data Cleaning and Imputation strategy

All string variables are converted to lower cases. Rows missing age, sex, and province information are removed. "Additional information" is tokenized, stop words are removed, and the domain in the source is parsed out. Empty strings are filled into additional information and sources that have missing values. Only "Taiwan" has country information missing, and "China" is filled into it. The province 'Caba' is replaced with 'Buenos Aires', and 'prefecture' is replaced with an empty string to keep consistency with location.csv. For countries categorized by district or region, "Not applicable" is filled if the province is missing. The remaining geographic information is imputed with reverse geo-encoding. The category "not available" is filled into the missing value of gender. The mean of the range has replaced the age range, and ages containing "months" have been parsed and divided by 12. Ages close to 0 are rounded to 0, and ages containing "+" or "-" are replaced by the closest mean of that value with either 0, 18, or 100. For example, "80+" is replaced to (80+100)/2 = 90. For

the age imputation, non-null values for 'long, 'lat', 'sex', 'date confirmation' are grouped by age and the mode of the group that best matches replace the missing value. If there are still missing values, it will take the mode of the overall age column to fill in the rest of the missing values. The intuition is to create a donor pool with similar feature values as the missing row and extract the most probable values. Two more columns are derived from age, one to measure the effect of averaging age ranges and filling missing values, and another to determine if any further generalization of age benefits performance. For date confirmation, date ranges are replaced with the first date in the range because it may represent when a patient is hospitalized until the patient leaves the hospital. The impact of such replacement will be small because there are only a few records with non-null date ranges. For missing values in the date confirmation column, the non-null values are grouped by long and lat, and the mode of the group that best matches the missing value's row is being filled in. The process repeats with province and country if no suitable match is found. Otherwise, we replace it with the mode of the entire column. The idea is that people in the same area are likely to be exposed to an outbreak simultaneously. Also, we turned date confirmation into a timestamp variable from a string variable, which will impose an ordering on the dates and the categorized data. This same assigning process is applied to the test dataset, except for removing low-quality entries in the initial step.

3.2 Dealing with Outliers

For this section, we removed outliers for various attributes. We removed the data from 141 rows containing ranges spanning more than 40 years for the age attribute. We felt recomputing this value in the imputation step will provide better quality to this age variable than a mean on such a massive range. Initially, we wanted to assume a normal distribution based on the histogram of age and purge the top 1% of data, but the values of age range from 0 (baby) to 106 after imputation, which all fall into a normal human lifespan. For the sex variable, we have 3 categorical values after imputation: male, female, and not available, with a reasonable split ratio between male and female. Since the source and additional information attributes have no specific structure, we conclude that no outliers are in those columns. The date confirmation column has no outliers because all of the day's, month's and year's values exist in a valid range.

That being said, there are some erroneous rows in the province column in which Finland is categorized by hospital districts. We did not consider them as outliers because removing them would remove all representation of Finland. We also detected a case for Ontario, China. This is a location that does not exist. So, we threw that out as well. Two rows concatenate 3 different areas by 'or' while other provinces under the same country refer only to a specific area; we removed the rows as they are considered the outliers. There are no outliers within the country columns after manual verification in the unique values. At first glance, the range of longitude and latitude are inside the appropriate range of -180 to 180 and -90 to 90. However, looking at the heat map, we realize that some cases are happening in the ocean. We decided to apply isolation forest onto the combination of long/lat to remove the outliers, but it was capturing too many valid pairs. We ended up using another package called global_land_mask, and it was able to detect long/lat pairs not located on land. Due to technological limitations, we still had to verify a few outlier pairs manually, but a total of 3 rows that were in the ocean were deleted manually.

3.3 Transforming Location Dataset

We took the US county data in locations.txt and aggregated it such that every county was combined into a single row for each state or territory. The latitude and longitude of the state were calculated

by taking the mean of the latitude and longitude data given by the counties, which should approximate the state's latitude and longitude reasonably well. For columns regarding the case, death, activity, and recovery count, we summed the appropriate column since the sum of cases for the data in all counties should be the sum of data for the state. We also recalculated the combined key for the state by setting it to form 'STATE, United States'. Given that the incidence rate is the prevalence of new cases per 100,000 population, we could sum over those columns in the county data. Finally, we recalculated the case-fatality rate with the new number of cases and deaths, but that was a straight calculation of (State Deaths)/(All Cases) * 100 for each respective state. One additional data cleaning step was taken when transforming the data because some rows had a negative number of active cases, which has no logical grounding. Therefore, these rows were filtered out and removed from location_transformed.csv.

3.4 Joining with Location Dataset

We used an adaptive technique to join our dataset. First, we did an exact match on longitude and latitude between the cases and location data. This helped to match approximately 11 rows that could not be joined alternatively due to the province's format, which differs from the province's format in the location_transformed.csv datafile. Then, we joined 'province, country' because approximately matching longitudes and latitudes will be much more challenging. Some cases may be registered within one province but have coordinates more similar to another. Because we imputed province and country data in an earlier step, this was an excellent matching technique.

However, some countries in our location_transformed.csv file contained no province information. After matching on 'province, country', we also matched on 'country' alone. Upon match completion, the remaining rows will fail to match by province and country as their data was not available in location.csv previous steps. Therefore, we aggregated the location.csv at the national level and took the mean for each column. We then matched it back to the missing values by the country column as well. Only 3 entries from Puerto Rico could not be matched back. Therefore, we imputed the values to the overall mean from the match result, representing the average value for a country.

4. Classification models

For this project, the final classification models that we decided to use are Catboost, Random Forest, XGBoost. The choice of Catboost is because there are categorical and text variables within the given and derived features. Catboost, a model which has its own internal coding techniques, such as target encoding, as well as the text processing feature, can deal with these variables without extra efforts. Additionally, since it does not apply one-hot encoding for all categorical variables, it can relax memory usage and increase accuracy as the dimensionality would not increase. The choice of random forest is because it leverages the power of multiple decision trees. Unlike decision trees, it chooses features randomly during the training process meaning it will not create a highly biased model while lowering the variance. Lastly, the choice of XGBoost is because we noticed that the deceased class often has very low accuracy and boosting, a method which leverages the errors in our prediction, may improve the performance of the model on this class. The only additional challenge for XGBoost was needing to encode our labels, but that was trivial. KNN had been attempted for milestone 2, but it is not time efficient and thus discarded. Other possible classifiers such as SVM and Naive Bayes are not feasible because it is a multi-class problem and the conditional independence assumption of Naive Bayes is not likely to occur given the dataset.

5. Initial Evaluation and Overfitting

5.1 Initial Evaluation

The train dataset has been split into 0.8:0.2 ratio as train and validation set. The metrics used are accuracy, precision, recall, and f1-score. Accuracy is not prioritized because there is a class imbalance problem. The metrics that we will be analyzing are the independent and macro average of the remaining metrics. Baseline models achieved ~70% Macro F1-Score in the Catboost & Random Forest model and only 58% in XGBoost model for the validation set, which indicates there is still room for improvement. By analyzing the metric in each outcome, we can see that prediction for non hospitalized is not a problem because F1-Score is nearly 100% in all models for either dataset, implying strong differences exist between the entries in this class compared to that of other classes. However, performance on the remaining outcomes are not as optimal, especially with a F1-Score of ~20% in deceased. From the confusion matrix in the /plots folder, we can infer that it is likely to be caused by insufficient data in the deceased label and entries from this label shares similar characteristics with entries under the hospitalized label, creating difficulties for the classifier to identify deceased entries correctly. A possible reasoning may be that patients are announced dead in the hospital. Similar findings are encountered in recovered labels being mistaken as hospitalized. This prompts the adjustment to regularization factor, weights and stratified sampling in order to counter these problems.

Catboost:

Training	precision	recall	f1-score	support
deceased	0.43	0.25	0.31	3583
hospitalized	0.83	0.83	0.83	99933
nonhospitalized	1.00	0.99	0.99	119402
recovered	0.74	0.76	0.75	70362
accuracy			0.87	293280
macro avg	0.75	0.71	0.72	293280
weighted avg	0.87	0.87	0.87	293280

Validation	precision	recall	f1-score	support
deceased	0.38	0.22	0.28	912
hospitalized	0.82	0.82	0.82	24862
nonhospitalized	0.99	0.99	0.99	30018
recovered	0.74	0.75	0.74	17528
accuracy			0.87	73320
macro avg	0.73	0.70	0.71	73320
weighted avg	0.87	0.87	0.87	73320

Random Forest:

Trainset

	precision	recall	f1-score	support
deceased	0.94	0.29	0.44	3583
hospitalized	0.81	0.89	0.85	99933
nonhospitalized	1.00	1.00	1.00	119402
recovered	0.81	0.73	0.77	70362
accuracy			0.89	293280
macro avg	0.89	0.73	0.77	293280
weighted avg	0.89	0.89	0.89	293280

Testset

	precision	recall	f1-score	support
deceased	0.52	0.13	0.21	912
hospitalized	0.80	0.88	0.84	24862
nonhospitalized	0.99	0.99	0.99	30018
recovered	0.78	0.71	0.74	17528
accuracy			0.87	73320
macro avg	0.77	0.68	0.70	73320
weighted avg	0.87	0.87	0.87	73320

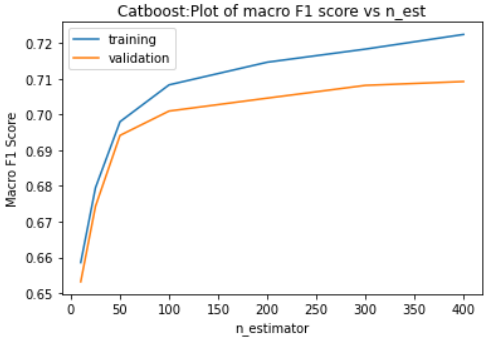
XGBoost:

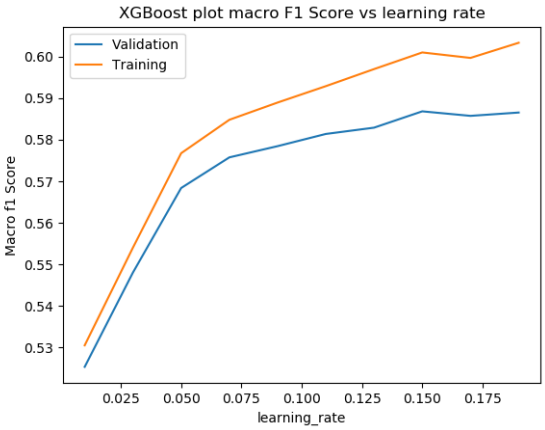
Training	precision	recall	f1-score	support
deceased	0.90	0.14	0.24	3583
hospitalized	0.67	0.87	0.76	99933
nonhospitalized	0.99	1.00	0.99	119402
recovered	0.67	0.41	0.51	70362
accuracy			0.80	293280
macro avg	0.81	0.60	0.62	293280
weighted avg	0.80	0.80	0.79	293280

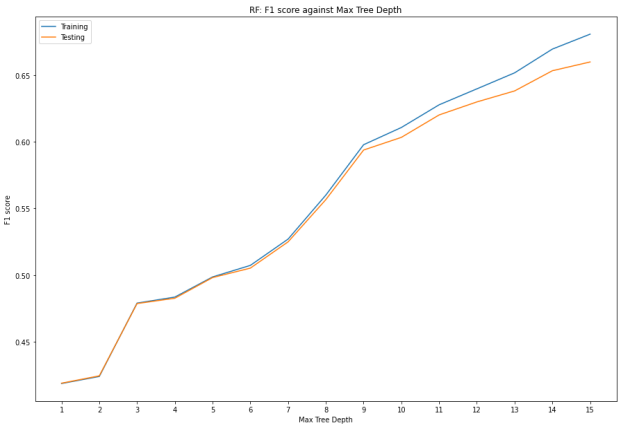
Validation	precision	recall	f1-score	support
deceased	0.54	0.07	0.13	912
hospitalized	0.67	0.86	0.75	24862
nonhospitalized	0.98	0.99	0.99	30018
recovered	0.65	0.40	0.50	17528
accuracy			0.80	73320
macro avg	0.71	0.58	0.59	73320
weighted avg	0.79	0.80	0.78	73320

5.2 Overfitting

To detect overfitting of the classifiers, we varied a parameter and built many models for each classifier. We then check “if there exist two models T and T’ such that T has better accuracy than T’ on training data but T’ has better accuracy than T on the test data”. If it does exist, then it indicates overfitting has occurred. The following pictures are results of Macro F1-Score on each model at different n_estimator for Catboost and Random Forest or learning rate for XGboost. As you can see, patterns in the training set always follow the validation set so there is no overfitting.







6. Hyperparameter tuning

The hyperparameter tuning is done on the train set that resulted from test train split. The validation set will be held out for comparison of the three models. The scoring function has been set to accuracy, overall recall (Macro), F1-score for deceased, Recall for deceased. Refit has been set to true to refit on the whole train set once the hyperparameters associated with the best F1-Score on deceased is discovered. Grid & Random search is configured using either 3 or 5 StratifiedKFold cross validation (CV). This will ensure the percentage of samples is preserved for each class in the CV. Complete csv files are in the results folder. In the tuning process, the range of the possible hyperparameters are selected around the hyperparameters from the baseline model because we know that the baseline model has reached ~70% Macro averaged F1-SCore for the validation set.

6.1 Grid Search

The GridSearchCV technique will exhaustively search over the specified hyperparameter values to find the best estimator on the desired metric. It builds models for all possible combinations of the given range and selects the best model it could build through this process.

6.1.1 Random Forest: 5 cv was set for the trade-off between the runtime and the model performance. The best combination of hyperparameters are with the criterion: gini,n_estimators: 20, max_features: 0.6. By turning the model, we know that using Gini criterion will get a better fl-score score than entropy criterion.

Top 5 hyparameter combinations

Hyperparameters			F1-score on 'deceased'	Recall on 'deceased'	Overall Accuracy	Overall Recall
Criterion	n_estimators	max_features				
Gini	20	0.6	0.214662	0.134238	0.872675	0.676462
Gini	20	0.8	0.213791	0.135074	0.872453	0.676642
Gini	100	0.8	0.213726	0.131726	0.872702	0.675982
Gini	50	0.6	0.212814	0.132004	0.872821	0.676117
Gini	100	0.6	0.211460	0.130328	0.872838	0.675763

6.2 Randomized Search

Randomized Search is another technique for optimizing hyperparameters of a given model. Rather than exhaustively searching through the possible combination of a given grid like in grid search, random search selects random combinations to train and model. The main advantage is that it allows us to go through a broader range of hyperparameters with less computation time than grid search. However, it does not guarantee that it will identify the best combination in the specified range.

6.2.1 Catboost: 5 CV is used and 60 parameters settings are sampled in the given range in each CV. The tuned hyperparameters are depth, learning rate, L2 regularization and n estimators as they are considered most important for tuning the model. The best combination of hyperparameters are depth: 6, l2 leaf reg: 0.146, learning rate: 0.3309, n_estimators: 339. The performance of the hyperparameter on the validation set will be presented in the next section. Auto class weight is set to true for each combination attempted in the search to prevent the class imbalanced problem.

Top 3 hyperparameter combinations

Hyperparameters	F1-Score on Deceased	Recall on Deceased	Overall Accuracy	Overall Recall(Macro)	Overall F1-Score(Macro)
{'depth': 6, 'l2_leaf_reg': 0.14642636433730596, 'learning_rate': 0.3309466283352577, 'n_estimators': 339}	0.260898532	0.202888743	0.863103519	0.689616091	0.701641342
{'depth': 8, 'l2_leaf_reg': 0.11002094791917921, 'learning_rate': 0.4934280568911248, 'n_estimators': 102}	0.259689513	0.205124549	0.862929624	0.689243469	0.700946658
{'depth': 6, 'l2_leaf_reg': 0.0258014780212234, 'learning_rate': 0.2990554581666141, 'n_estimators': 395}	0.258441616	0.202608635	0.86403437	0.689963557	0.701706398

6.2.2 XGBoost: 5 CV is used here totalling 100 fits. Hyperparameter settings are sampled from learning_rate between [0.01,0.05], max_depth between [6,10] and min_child_weight between [0.2,1]. XGBoost also used the randomCV approach. It found the best combination to be learning rate: 0.03, max_depth: 10, min_child_weight: 0.2. Of course, XGBoost has many more parameters, but these were most impactful, so they were chosen for tuning.

Top 3 hyperparameter combinations

Hyperparameters	F1-Score on Deceased	Recall on Deceased	Overall Accuracy	Overall Recall(Macro)	Overall F1-Score(Macro)
{'min_child_weight': 0.2, 'max_depth': 10, 'learning_rate': 0.03}	0.119254977292923	0.065582073038654	0.790459629023459	0.570405211805655	0.573402736044591
{'min_child_weight': 0.6, 'max_depth': 10, 'learning_rate': 0.03}	0.11598557732577	0.063628713681307	0.790330060010911	0.569892916070043	0.572625943934563
{'min_child_weight': 0.4, 'max_depth': 9, 'learning_rate': 0.05}	0.111480051323448	0.061115526362949	0.790602836879433	0.570307216356635	0.573414340334194

7. Results

The result of the tuning did not significantly improve the model on the validation set and it even worsened the model for XGBoost on the validation set even after a runtime of 7 hours. For Catboost and Random Forest, there are no changes on the F1-Score of nonhospitalized, recovered nor hospitalized on the validation/training set, which is understandable because the best model is optimized toward the best F1-Score on deceased. Additionally, the deceased label does not contain many entries, so the impact from reducing the false negative or false positive in this class may not be well reflected in the score of other outcomes. However it surprised us that even after the parameter tuning and the assignment of the weights for each class, the increase in f1-score is only 1% for both Catboost and Random Forest. The only metric that experienced significant improvement is precision for the deceased class in the Catboost model. Other metrics such as overall recall, recall for deceased or accuracy did not undergo any improvement greater than 1%. From the confusion matrix, we can see that the deceased label continues to be misclassified as hospitalized. In fact, there are 20 more deceased entries that got classified as hospitalized than the baseline model. However, the overall number of false positives decreased by a large amount hence increasing precision, consistent with our classification report.

The decay in performance for the XGBoost may be due to the relatively wide search space for each hyperparameter, paired with the randomized method, and too few combinations to find anywhere near the optimal combination of hyperparameters. This results in a decrease in most metrics, except a one percent increase in the f1-score of the hospitalized label. Unfortunately, increasing the combinations attempted is very costly, but methods for improvement to this result are mentioned in the "future work" section. So, the results for XGBoost were higher with the initial human-selected values, likely because they were chosen off averages of what works best in the model, rather than an attempt at randomly searching a range of hyperparameters for the most optimal for our dataset.

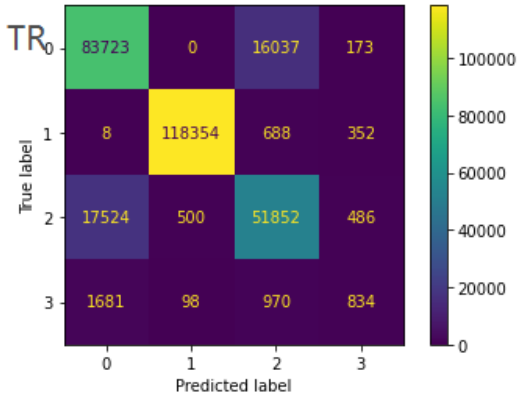
Catboost:

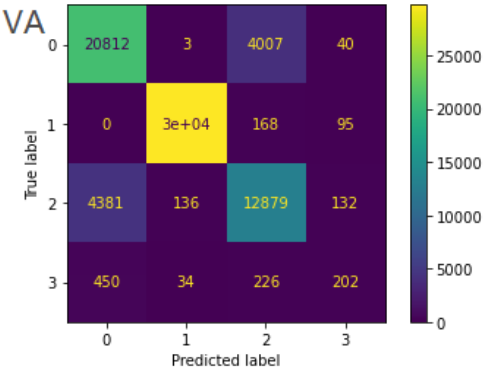
Train

	precision	recall	f1-score	support
deceased	0.45	0.23	0.31	3583
hospitalized	0.81	0.84	0.83	99933
nonhospitalized	0.99	0.99	0.99	119402
recovered	0.75	0.74	0.74	70362
accuracy			0.87	293280
macro avg	0.75	0.70	0.72	293280
weighted avg	0.87	0.87	0.87	293280

Validation

	precision	recall	f1-score	support
deceased	0.43	0.22	0.29	912
hospitalized	0.81	0.84	0.82	24862
nonhospitalized	0.99	0.99	0.99	30018
recovered	0.75	0.73	0.74	17528
accuracy			0.87	73320
macro avg	0.75	0.70	0.71	73320
weighted avg	0.87	0.87	0.87	73320





Random Forest:

Train

	precision	recall	f1-score	support
deceased	0.92	0.29	0.45	3583
hospitalized	0.81	0.89	0.85	99933
nonhospitalized	1.00	1.00	1.00	119402
recovered	0.81	0.73	0.77	70362
accuracy			0.89	293280
macro avg	0.89	0.73	0.77	293280
weighted avg	0.89	0.89	0.89	293280

Validation

	precision	recall	f1-score	support
deceased	0.47	0.14	0.22	912
hospitalized	0.80	0.87	0.84	24862
nonhospitalized	0.99	0.99	0.99	30018
recovered	0.78	0.71	0.74	17528
accuracy			0.87	73320
macro avg	0.76	0.68	0.70	73320
weighted avg	0.87	0.87	0.87	73320

XGBoost:

Train	precision	recall	f1-score	support
deceased	0.86	0.09	0.16	3583
hospitalized	0.65	0.91	0.76	99933
nonhospitalized	0.98	1.00	0.99	119402
recovered	0.69	0.33	0.44	70362
accuracy			0.79	293280
macro avg	0.80	0.58	0.59	293280
weighted avg	0.80	0.79	0.77	293280

Validation	precision	recall	f1-score	support
deceased	0.71	0.07	0.12	912
hospitalized	0.65	0.91	0.76	24862
nonhospitalized	0.98	0.99	0.99	30018
recovered	0.69	0.33	0.44	17528
accuracy			0.79	73320
macro avg	0.76	0.57	0.58	73320
weighted avg	0.80	0.79	0.77	73320

8. Conclusion

The main metric that we will base off our comparison is the F1-Score on deceased as that is the minority class and the metric that calculates its score based on both false negative and false positive, consistent with the objective of this milestone. Other metrics such as recall on deceased and macro average for f1-score will be used to break ties between models. From the evaluation metrics of the 3 models, Catboost has the best performance with a F1-Score on deceased of 29%, Random Forest with the second best performance of 22% with F1-Score on deceased, and lastly XGBoost with only 12%. The models mentioned above are built with the best hyperparameters discovered during the tuning phase. This ranking is true on other metrics as well. We also compared the amount of entries from each class for the prediction on the test set. Out of 46500 total entries in the test set, there are 496 (1%) patients that are classified as deceased in Catboost, 2220 (5%) in Random Forest and only 97 (0.2%) in XGBoost. If we assume that the same distribution exists in the test set, prediction with Catboost has the most probable amount of entries for the deceased class as well. Therefore, we can conclude that Catboost is the best fit.

As for the advantages of Catboost, the internal encoding of the categorical variables represents the information much more concisely than the one-hot encoding method that was used in Random Forest and XGBoost. Since, those models could not directly handle categorical variables. This resulted in the increased correctness of the deceased class in the Catboost model. Random Forest has the advantages of handling large data sets with high dimensionality, estimating missing data and maintaining accuracy. Also, it gives variable importance which helps in understanding impact of variables. However, it does not perform very well when there are rare outcomes as it is based on bootstrap sampling. In this project, this limitation is demonstrated with the large gap of difference in f1 score on 'deceased' between training data and testing data. The problem with the XGBoost model is that it has 12 boosting parameters. This made it very challenging to choose which three to tune for this milestone. Even though the most pertinent ones were chosen, it failed to yield the best result. This problem would have only been made worse if more parameters were considered, as the computational cost of checking so many combinations of hyperparameters with 5 fold CV is incredibly high. But the model has its benefits, since it has built in regularization, it helps reduce overfitting. This is exemplified by the graph in our overfitting section, that shows XGBoost did not exhibit any.

9. Prediction on test dataset

To predict the test data, we perform the same data cleaning and imputation process that were applied on the training data. However, we do not apply any outlier detection nor remove any entry because we want to preserve the full amount of observation in the test dataset. Once the test data is processed, the data is fed into each of the models and a text file of outcome will be produced corresponding to its index.

10. Lessons learnt and future work

Future work we could do to improve this project is to collect more data through the open source. We have learnt from milestone 1, how to best clean, induce values and generally preprocess our datasets. In milestone 2, we learnt how to detect overfitting in the training data, and strategies to mitigate this. Namely, ranges of hyperparameters to avoid. We also learnt that although our dataset had a relatively low dimensionality, even after one-hot encoding (only 17), it was still too high to reasonably pursue a knn model. The computational time made it very challenging to attempt various hyperparameters, as the model took over an hour to predict on the training data for just a single combination of hyperparameters. Which is why we changed it to a XGBoost model for the final milestone. In this final milestone, we learnt how to best tune our hyperparameters with both grid and random grid approaches to cross-validation.

SMOTE: The distribution of the predicted outcome is skewed in our classification, only 1 % of our data is labeled deceased. Therefore it is a challenge to get a high f1-score on the deceased outcome data. The Synthetic Minority Over-sampling Technique was incorporated into the Catboost model to increase the score on the deceased label, but it decreased the performance instead. Efforts can be made to adjust the oversampling ratio such that we can take advantage of the synthetic data. Undersampling can be attempted as well on the non hospitalized entries as well to balance the distribution of the class as the metric on this class is nearly 100%.

XGBoost: This model has a lot of intricacies, and an innate ability to increase accuracy with a minority class. Unfortunately, we witnessed a small decrease in performance of this model after tuning the hyperparameters. Further work to improve this model would be done in a number of ways. First of all, tuning more hyperparameters. Many were held static, or default when the model was tuned, due to the high cost of tuning too many parameters at once. However, tuning a parameter such as scale_post_weight could improve the performance on the minority class (which is what we care about!). Except, it is less critical than the chosen tuned parameters, so it would be an additional task to explore. Second of all, if hyperparameters were tuned more rigorously as to converge closer to the best parameters for the model. This would also amount to a much higher computational cost, but it would likely vastly improve the results.

11. References

1. Pedregosa et al. “Scikit-learn: Machine Learning in Python”, JMLR 12, pp. 2825-2830, 2011. https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

2. Raghav RV. Multi-metric evaluation documentation. Retrieved from https://scikit-learn.org/stable/auto_examples/model_selection/plot_multi_metric_evaluation.html

3.Banerjee, Prashant. “A Guide on XGBoost hyperparameters tuning”, kaggle, 2020, https://www.kaggle.com/prashant111/a-guide-on-xgboost-hyperparameters-tuning

4. Catboost Parameter Tuning documentation. Retrieved from https://catboost.ai/docs/concepts/parameter-tuning.html

12. Contributions

Amos: M1 - 1.2 / 1.3 / 1.5 + Report | M2 M3 - Catboost + Report
Bianca: M1 - 1.4/1.5(unused) + Report | M2 - Knn + Report | M3 - XGBoost + Report
Timothy: M1 - 1.1/1.6 +Report | M2, M3 - Random Forest + Report