Anmol Saini ahsaini@sfu.ca
Jason Cao yjc2@sfu.ca
Jograj Sidhu jograjs@sfu.ca

Mining COVID-19 Data

CMPT 459: Milestone 3

I. Problem Statement

The objective of the project is to build an optimal classification model and use it to accurately classify the health outcomes (hospitalized, non-hospitalized, recovered, deceased) of COVID-19 patients from a publicly available COVID dataset. The process of predicting the outcomes consists of three phases: data preprocessing, building baseline models along with finding the appropriate metrics for model evaluation, and optimizing the baseline models through hyperparameter tuning and choosing the best model through comparisons of their classification performance. The finalized prediction model should be robust thus capable of generalizing unseen data. Its effectiveness in the overall predictive power can be assessed through combinations of high accuracy score, precision, recall and f1-score. More importantly, it is crucial to predict the deceased with as few false negatives as possible while keeping the number of false positives relatively low.

II. Dataset Description and Exploratory Data Analysis

Publicly available COVID-19 datasets used for this project consists of:

i. *cases_test.txt*, an anonymized patient-level file containing 367,636 patients basic information and their associated COVID health status (i.e., outcome label).
ii. *cases_train.txt,* an anonymized patient-level file containing basic patient information of 46,500 patients, but without any outcome label.
iii. *location.txt,* file containing COVID statistics at a population-level for every country and province pair. There are 3,954 rows in this dataset.

Multiple observations are made to understand the datasets being used. First, recognizing the quantity of NULL values of each feature in all three datasets is required to know the scale and type of imputation that must be done in data preprocessing. Table 1 shows the missing values of the data. As can be seen, the columns ‘age’ and ‘sex’ from the cases datasets are essential information which contain many NULL values. Figure 1 shows the distribution of gender from

Table 1. Missing Values in the three datasets.

Cases_train		*Cases_test*		*Location*	
age	209265	age	25516	Province_State	168
sex	207084	sex	25199	Country_Region	0
province	4106	province	573	Last_Update	0
country	18	country	0	Lat	80
latitude	2	latitude	0	Long_	80
longitude	2	longitude	0	Confirmed	0
date_confirmation	288	date_confirmation	49	Deaths	0
additional_information	344912	additional_information	43490	Recovered	0
source	128478	source	16903	Active	0
outcome	0	outcome	46500	Combined_Key	0
				Incidence_Rate	80
				Case-Fatality_Ratio	0

the training dataset, with 56% of *Unknown* gender. After converting the different types of values in the age column to the integer type, the histogram in Figure 2 below shows the frequency of COVID-19 cases for each age range, with higher cases of 25-50 aged population.

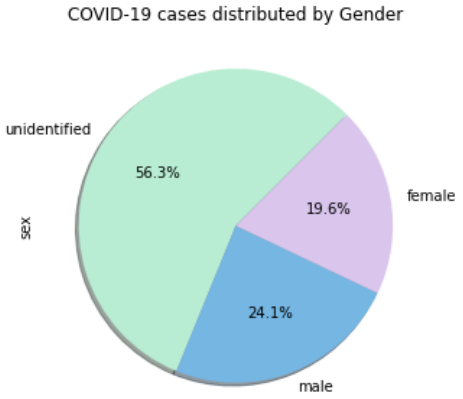

Figure 1. Gender Distribution

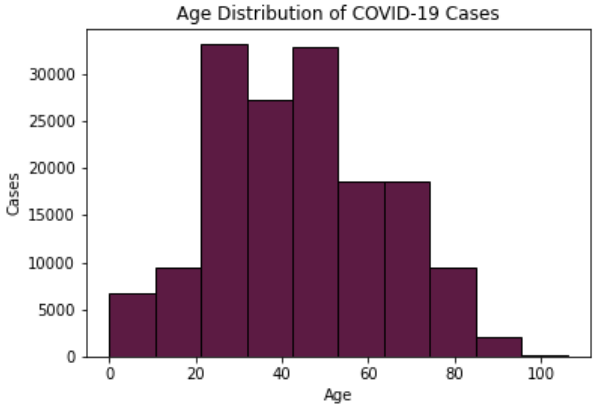

Figure 2. Age Distribution Histogram

Furthermore, to depict the cases in a geographical sense, the 'latitude' and 'longitude' values from the training dataset are used to create a visual of the dissemination of COVID-19 cases around the world, in Figure 3. Many countries affected by COVID-19 are easily represented by this map.

Figure 3. Total Cases in the world plotted by latitude and longitude values from cases_trained.

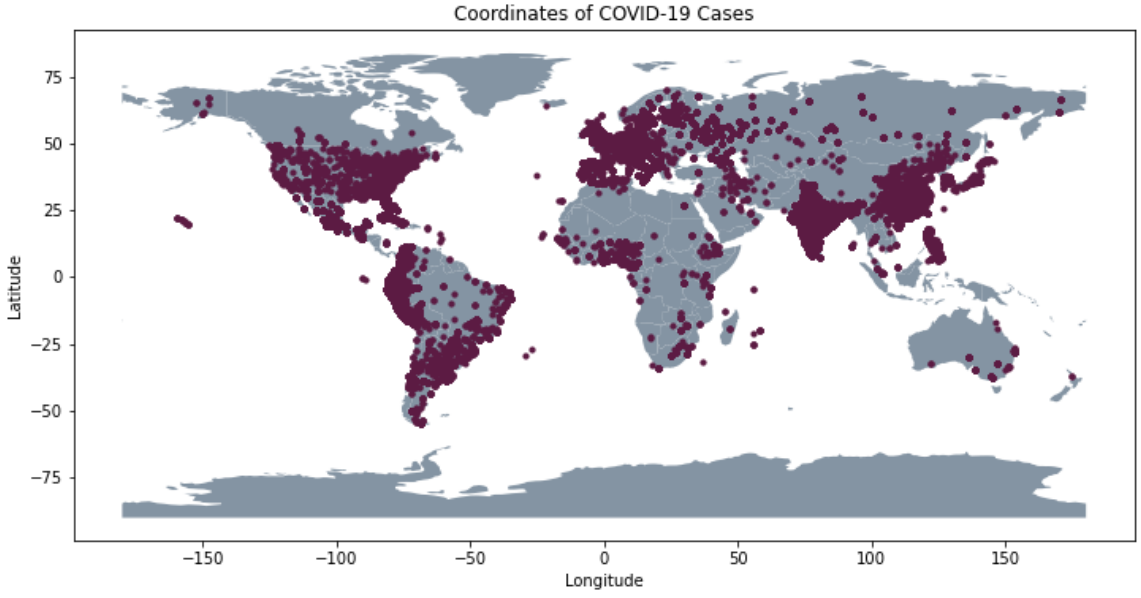

Figure 4 shows the distribution of outcomes from the training data: *nonhospitalized*, *hospitalized*, *recovered*, and *deceased*. This helps to recognize the imbalance in the classification labels. It is understandable, since the reports of less deceased cases and more people recovered and non-hospitalized is preferred than the opposite. However, imbalanced classes make it difficult to predict the minority class (*deceased*), which can heavily impact the correctness of the models.

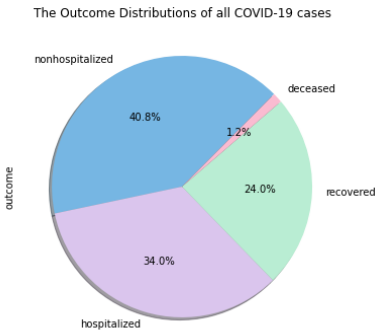

Figure 4. Pie chart of the outcome labels.

III. Data preparation

III.1 Data Cleaning and Missing Values Imputation

There are features with large amounts of missing values such as *age, sex, province, additional information* and *source* from the training and test sets, and *province_states* missing from the location set. To impute the missing values in the *age* column, first the existing values need to be converted to the same data type format which is done by Regex. Leading and trailing characters are removed, age in months are converted to years, and range values are replaced with the average of the range. Then, the mean of the entire column is used to impute the empty rows. Since there is no way to know whether the patient is a Male or Female from the other information, the missing values in the Gender column are replaced with "Unknown" string. The missing Date Confirmation values are replaced with "20.09.2020" string since it is the last date the most recent data was acquired. All values are then converted into integer data type for easy usability for the models. For the columns *additional information* and *source,* they don't provide much substantial data so the missing values are filled with 0 to represent "missing", and the non-missing values are replaced with 1. The geocoding library, GeoPy, is used to find the missing province and country names from all three datasets by using the longitude and latitude values and reverse geolocating.

Table 2. Outlier Detection with the IQR method and the actions taken.

Attributes	Interquartile Range			Potential Outliers	Actions Taken
	IQR Value	Lower Bound	Upper Bound		
Age	29	-16	101	0	Age values are between 0 to 99, which are quite reasonable; so no action taken.
Latitude	14.895	-9.660	49.919	84142	All values are within the actual range for latitudes (-90 to 90) and longitudes (-180 to 180); so no action taken.
Longitude	77.352	-116.172	193.238	295	
Lat	8.888	19.938	55.491	415	
Long_	18.972	-125.069	-49.181	530	
Confirmed	1992	-2851	5117	639	All 3 columns are without missing values and are used to derive other column values (Active & Case.Fatality_Ratio); so no action taken.
Deaths	47	-70	119	609	
Recovered	0	0.0	0.0	614	
Active	1339	-1895	3462	543	The 6 negative outliers were summed into their corresponding group total via province-country group-by aggregation.
Incidence_Rate	1479.089	-1599.730	4316.628	155	No individual values ever fall below 0; hence no action taken.
Case-Fatality_Ratio	2.369	-2.916	6.560	246	

III.2 Outlier Detection

Since Z-score can be easily influenced by extreme values and only works well with normal distribution, Interquartile Range (IQR) is used, for it is robust to any specific distribution and lacks the sensitivity to the presence of extreme outliers. To systematically spot potential outliers of a large dataset, IQR 25th and 75th quartile values and an adjustment factor are combined to define statistical boundaries on the numeric columns of training and location dataset. The results of outlier detection are displayed in Table 2 above. While some suspected

outliers are quite obvious, the methods to handle them require a much thorough examination of the context:

i. There are no outliers for nominal values.
ii. Numeric ranges (min and max) reveal no significant outliers except 6 negative values from Active attribute, which are subsequently included into their respective province-country group sums.
iii. Latitudes and longitudes outside the acceptable locations (e.g. coordinates pointing to the middle of the ocean) have their corresponding country and province attribute values replaced with "Unknown".

III.3 Transformation

Prior to merging, records of the same province of the same country are aggregated into a single record. Confirmed, Death, and Recovered values of each province-country group can be summed to account for the totality of Confirmed, Death, and Recovered cases from each group. Active values (Confirmed - Death - Recovered) and Fatality ratios (Death / Confirmed * 100) for each group can be re-calculated for each aggregated group. Incidence rates from the same province-country group are averaged rather than recalculated because population data, not present in the current dataset, are required.

Post-merging transformation involves converting string values into numerics. While one-hot encoding has the benefit of not weighting a value improperly by creating more columns to the dataset, it can cause the number of columns to expand exponentially if there are many unique combinations of values such as the ones from province and country columns. As such, *label-encoding* is used to simply represent each unique value in a column to a corresponding number.

III.3 Dataset Merging

To subsequent joining of location set with training data, and location with testing data, the *province* and *country* attributes in each dataset are combined to create a *province-country* key attribute. For provinces that are "Unknown," the rows of the respective country are aggregated and used as the key for joining the datasets. Faulty merge due to name inconsistencies (from province or country) are fixed by standardizing the names. For example, one data set had a country named "North Korea" while another had it named "Korea North".

IV. Classification models

The chosen classifiers are K-Nearest Neighbors (KNN), Random Forests (RF) and LightGBM. KNN is very versatile and easy to implement. Unlike RF and LightGBM, KNN does not have as many hyperparameters that need to be tuned. Training KNN is fast since it does not need training data points for model building. The base KNN classifier is trained on the two most influential parameters: n_neighbors = 9 and metric = 'distance'. The n_neighbors represent the number of points that classify the target point and the metric parameter makes it so closer points have more influence in the decision of the classification. RF builds multiple decision trees to obtain more accurate and stable prediction while effective at reducing overfitting by random sampling to minimize the correlation thus the variance of decision trees. RF base classifier is trained on

max_depth = 22 to specify the maximum depth of decision trees. LightGBM is fast and has better accuracy than other boosting algorithms. It handles large dataset and is proficient at dealing with categorical and NULL values without prior encoding and imputations. As a base classifier, LightGBM specifies the maximum number of leaves for the trees to be 100.

V. Initial evaluation and overfitting

Accuracy scores for all three baseline models are comparable. Validation scores are lower by at most 2% than the training scores, indicating no sign of overfitting.

Table 3. Accuracy scores on baseline models.

Models	**Training Accuracy**	**Validation Accuracy**
KNN	0.88185	0.86416
Random Forest (RF)	0.87762	0.86534
LightGBM	0.87861	0.87062

However, since the dataset is heavily imbalanced, the resulting accuracy metric contains bias towards the minority class. Specifically, the proposed classification methods will simply ignore and in turn have poor performance when classifying the minority class. Subsequent evaluation metrics (precision, recall and F-score) are therefore utilized to assess model performance at a granular scale, depicting moderate to high proficiency when classifying hospitalized, non-hospitalized and recovered cases but consistently poor performance on deceased cases for both training and validation. In particular, low F1 scores in training (stemming from low recall) and very low F1 scores in validation (stemming from low precision and very low recall values).

Table 4. The precision, recall, and F1 scores on outcome labels for the baseline models.

Label	**Model**	**Training**			**Validation**		
		Precision	*Recall*	*F1-Score*	*Precision*	*Recall*	*F1-Score*
deceased	*KNN*	0.88	0.33	0.48	0.47	0.11	0.18
	RF	0.98	0.24	0.39	0.73	0.11	0.19
	LGBM	0.93	0.20	0.33	0.66	0.11	0.18
hospitalized	*KNN*	0.81	0.87	0.84	0.79	0.86	0.82
	RF	0.79	0.88	0.84	0.78	0.88	0.83
	LGBM	0.80	0.89	0.84	0.79	0.88	0.83
non hospitalized	*KNN*	1.00	1.00	1.00	0.99	0.99	0.99
	RF	1.00	1.00	1.00	0.99	0.99	0.99
	LGBM	1.00	1.00	1.00	0.99	0.99	0.99
recovered	*KNN*	0.79	0.73	0.76	0.76	0.69	0.72
	RF	0.79	0.70	0.74	0.77	0.67	0.72
	LGBM	0.8	0.70	0.75	0.78	0.68	0.73

To determine how well-separated the four classes are under the three chosen base models, confusion matrices are used to visually examine the competency of each model. Overall, all three base models show high competency in classifying non-hospitalized cases. They all have trouble distinguishing hospitalized cases and recovered cases. Lastly, they are unable to effectively classify deceased cases. Figure 5 below is a confusion matrix for a KNN classifier.

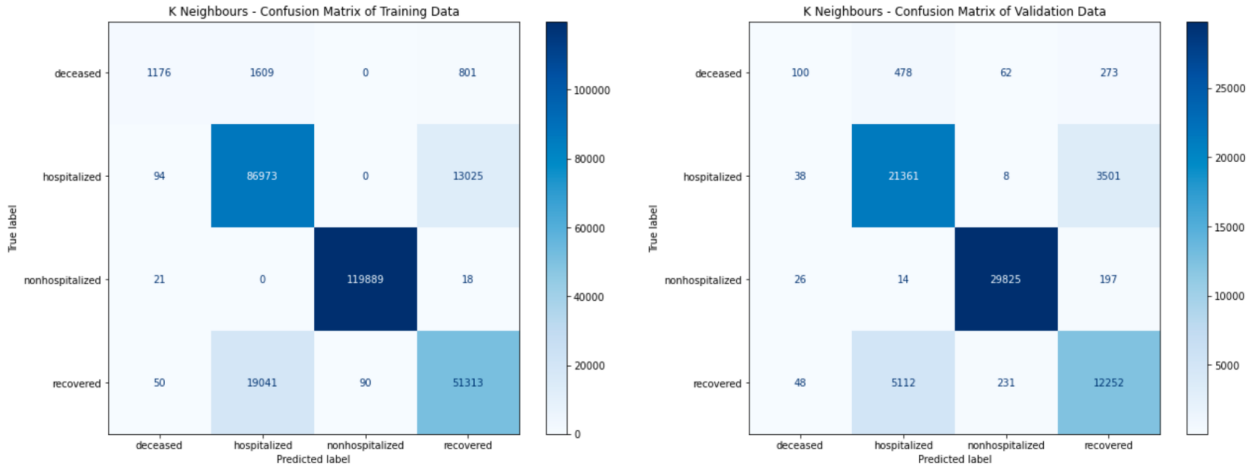

Figure 5. A confusion matrix of the KNN model for training and validation data.

VI. Hyperparameter tuning

Hyperparameter tuning is done to improve the performance of baseline models. Our approach starts with partitioning the training dataset into 80% for training and 20% for validation. Utilizing the results of accuracy curves from milestone 2, three hyperparameters along with a short list of their corresponding values are chosen for tuning baseline models. Scikit-learn's GridSearchCV receives those input values and creates an exhaustive list of every combination of hyperparameter values such that each combination can be used to construct a classification estimator. GridSearchCV allows for input of more specific values chosen from the accuracy curves from Milestone 2 thereby minimizing overfitting.

Subsequent assessments for each estimator requires a k-fold cross validation which involves in resample from the training set k times to create k variant estimators. As a variant estimator is built, it is evaluated on the training set and validation set. These k variant evaluations are then averaged to produce overall performance scores (accuracy, precision, recall, f1) for the estimator.
Given how large the size of the original dataset, iteratively running a 3-fold CV for RF/LightGBM and a 2-fold CV for KNN, and a selective range of hyperparameters are done to balance the trade-off between program running time and model performance.

KNN Parameter Values	**RF Parameter Values**	**LightGBM Parameter Values**
n_neighbors: 9, 30, 50	max_depth: 29, 30	num_leaves: 60, 90, 120
weights: uniform, distance	n_estimators: 200, 250	n_estimators: 100, 200, 300
metric: euclidean	max_features: 10, 12	min_data_in_leaf: 60, 80

Table 5. Possible hyperparameter values for their respective models.

VII. Results

Prior to hypertuning the models, the 80% training dataset split is oversampled to reduce the class imbalance. It is a partial oversampling where the sampling strategy sets a specified number for the samples of each label, as such:

```
strategy={'deceased':6000,'recovered':100500,'hospitalized':109000}
```

6000 is chosen as the deceased sample size because it is the threshold value that significantly changes the F1 score. Figures 6, 7, and 8 show the results from hyper-tuning parameters for the KNN, Random Forest, and LightGBM Classifiers, respectively.

Figure 6. KNN Hyperparameter Tuning Results

	n_neighbours	weights	metric	f1_deceased	rank_f1_deceased	recall_deceased	overall_accuracy	overall_recall
0	9	distance	euclidean	0.225773	1	0.165000	0.841930	0.676369
1	9	uniform	euclidean	0.181892	4	0.125333	0.833997	0.660764
2	30	distance	euclidean	0.214818	2	0.133167	0.846298	0.672542
3	30	uniform	euclidean	0.133662	5	0.073833	0.825817	0.642344
4	50	distance	euclidean	0.204695	3	0.125333	0.844873	0.669739
5	50	uniform	euclidean	0.102752	6	0.055333	0.816664	0.630860

Figure 7. Random Forest Hyperparameter Tuning Results

	max_depth	n_estimators	max_features	f1_deceased	rank_f1_deceased	recall_deceased	overall_accuracy	overall_recall
0	29	200	10	0.226061	6	0.198833	0.840699	0.683288
1	29	250	10	0.224963	8	0.198000	0.840788	0.683150
2	29	200	12	0.227278	4	0.198667	0.840881	0.683345
3	29	250	12	0.228944	1	0.200333	0.840917	0.683863
4	30	200	10	0.227830	2	0.201000	0.840869	0.683907
5	30	250	10	0.225613	7	0.198333	0.840598	0.683154
6	30	200	12	0.226454	5	0.198167	0.840750	0.683204
7	30	250	12	0.227327	3	0.198833	0.840768	0.683313

Figure 8. LightGBM Hyperparameter Tuning Results

	num_leaves	n_estimators	min_data_in_leaf	f1_deceased	rank_f1_deceased	recall_deceased	overall_accuracy	overall_recall
0	60	100	60	0.192803	15	0.150833	0.842291	0.673520
1	90	100	60	0.197291	13	0.161500	0.845296	0.678477
2	120	100	60	0.190844	16	0.169000	0.841060	0.676597
3	60	200	60	0.205710	8	0.167667	0.848242	0.682277
4	90	200	60	0.203626	9	0.178167	0.846033	0.682819
5	120	200	60	0.206582	5	0.188667	0.844327	0.683822
6	60	300	60	0.202030	10	0.180833	0.843558	0.681400
7	90	300	60	0.211058	3	0.193000	0.845284	0.685610
8	120	300	60	0.218102	1	0.201500	0.845719	0.687910
9	60	100	80	0.190832	17	0.147167	0.841951	0.672530
10	90	100	80	0.195525	14	0.159500	0.844545	0.677384
11	120	100	80	0.187688	18	0.164333	0.840505	0.675100
12	60	200	80	0.205771	7	0.165500	0.847949	0.681568
13	90	200	80	0.199722	12	0.178833	0.842610	0.680126
14	120	200	80	0.205920	6	0.186833	0.844345	0.683415
15	60	300	80	0.202009	11	0.179667	0.843621	0.681197
16	90	300	80	0.209129	4	0.191500	0.844706	0.684781
17	120	300	80	0.214152	2	0.198000	0.845547	0.686918

From the results above, some patterns of the parameter values were observed. For example, KNN has consistently low F1 scores when the weights parameter was set to 'uniform'. When the parameter was set to 'distance', the F1 score was more comparable to the other two models chosen.

VIII. Conclusion

For all models, precision significantly decreases when recall is slightly improved whenever the value of deceased is oversampled to higher than 6000. Experimental evidence shows 6000 to be the threshold value that tips the scales between precision and recall. When oversampling at the bound of threshold value in the training data, the precision of deceased in the validation data scores of the Random Forest model remains around 58% but with recall at only 12%. However, when oversamples at slightly beyond the threshold value (e.g., deceased = 6001), recall increased to 43% with a significant decrease to 7% for precision which greatly decreased the F1 score as well.

Table 6. Scores on the Training and Validation sets of the best tuned models.

	Training	**Validation**
KNN	Accuracy score: 0.86308 precision recall f1-score deceased 0.84 0.36 0.51 hospitalized 0.80 0.80 0.80 nonhospitalized 1.00 1.00 1.00 recovered 0.78 0.80 0.79 accuracy 0.86 macro avg 0.85 0.74 0.77 weighted avg 0.86 0.86 0.86	Accuracy score: 0.85578 precision recall f1-score deceased 0.40 0.12 0.19 hospitalized 0.82 0.79 0.80 nonhospitalized 0.99 0.99 0.99 recovered 0.70 0.76 0.73 accuracy 0.86 macro avg 0.73 0.67 0.68 weighted avg 0.86 0.86 0.85
RF	Accuracy score: 0.87446 precision recall f1-score deceased 0.97 0.38 0.55 hospitalized 0.81 0.83 0.82 nonhospitalized 1.00 1.00 1.00 recovered 0.80 0.81 0.80 accuracy 0.87 macro avg 0.89 0.75 0.79 weighted avg 0.88 0.87 0.87	Accuracy score: 0.86333 precision recall f1-score deceased 0.56 0.13 0.21 hospitalized 0.82 0.82 0.82 nonhospitalized 0.99 0.99 0.99 recovered 0.72 0.75 0.74 accuracy 0.86 macro avg 0.77 0.67 0.69 weighted avg 0.86 0.86 0.86
LGBM	Accuracy score: 0.87608 precision recall f1-score deceased 0.95 0.31 0.47 hospitalized 0.82 0.82 0.82 nonhospitalized 1.00 1.00 1.00 recovered 0.80 0.82 0.81 accuracy 0.88 macro avg 0.89 0.74 0.77 weighted avg 0.88 0.88 0.87	Accuracy score: 0.87149 precision recall f1-score deceased 0.52 0.12 0.19 hospitalized 0.83 0.82 0.83 nonhospitalized 0.99 0.99 0.99 recovered 0.73 0.78 0.75 accuracy 0.87 macro avg 0.77 0.68 0.69 weighted avg 0.87 0.87 0.87

When assessing the predictive performance of these three models, they all have modest to high accuracy, precision, recall and f1-score on classifying hospitalized, non-hospitalized and recovered cases for the training dataset and similar scores for the validation dataset. Notwithstanding their good performances, they all seem to suffer from low precision, recall and f1-score on classifying deceased cases. In terms of running time performance, both tree algorithms, despite how time-consuming it is to select the appropriate hyperparameters, run much faster than KNN.

IX. Prediction on test dataset

Random Forest classifier has the best results. It has the highest precision, recall and F1 scores compared to the other two models when predicting the test data. Predictions of the outcomes on the COVID-19 test dataset are shown in a pie chart distribution in Figure 9. As shown, the model predicts that 222 cases (0.48%) of the test data are considered deceased, 15164 cases (33%) predicted as hospitalized, 20064 cases (43%) as nonhospitalized, and 11048 cases with outcome of recovered. This predicted outcome distribution is very similar to the distribution of outcomes on the original training dataset from Figure 4, indicating a logical classification on the test data.

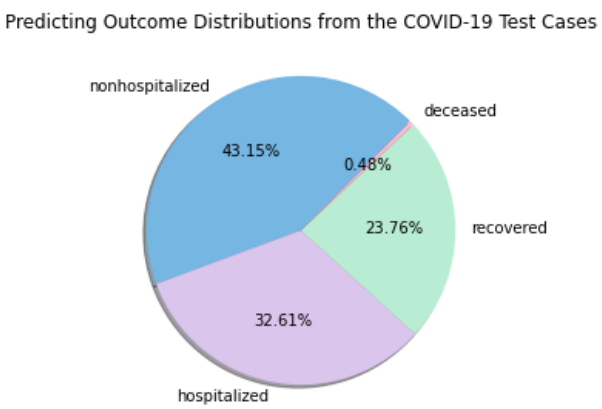

Figure 9. Predicted outcomes distribution

X. Lessons learnt and future work

It is very challenging to perform classification on an imbalanced dataset with an important class being exceedingly underrepresented (only 1.2% deceased). Feature selection was explored which did not seem to improve any of the scores. Multiple sampling techniques (SMOTE, AYDE, SMOTEENN, random oversampling, random undersampling and mix of over and undersampling) were subsequently conducted with slight improvement when applied only to the targeted class. When sampling techniques were applied to the entire dataset, results actually worsened. Consequently, the process of training a robust ML algorithm needs to be an iterative endeavour that has to be done repeatedly from preprocessing, to training then validation.

For future improvements, the first step would be to revisit the preprocessing stage and impute the data better, which could significantly improve the results. Further techniques to increase the recall and keep the precision high on the models will be investigated. With these improvements, it will be possible to make good predictions on new test datasets.

XI. References

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. https://scikit-learn.org/stable/index.html

LightGBM. (n.d.). *LightGBMClassifier.* https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html#lightgbm.LGBMClassifier

Brownlee, Jason (January 15, 2020). *Random Oversampling and Undersampling for Imbalanced Classification* https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Mandot, Pushkar. (August 17, 2017). *What is LightGBM, How to implement it? How to fine tune the parameters?* Medium. https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc

LightGBM. (n.d.). *Parameters Tuning — LightGBM 3.2.1.99 documentation.* https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html

XII. Contributions

Anmol: EDA, imputations, joining datasets, built and tuned LightGBM classifier, organized notebook and project github, and helped with writing reports.

Jason: outlier detection, transformation, built and tuned Random Forest classifier and helped with drafting reports.

Jograj: imputations, data merging, built and tuned K-Nearest Neighbors classifier and helped with writing milestone reports.