Milestone 2 report

2.1 - Splitting Dataset

To split the dataset, we passed in the processed train dataset and used 'sklearn.model_selection.train_test_split' function to split the data into a 80-20 ratio.

2.2 - Building Models

LightGBM Classification Model:

For the boosting tree algorithm, we chose the Python's LightGBM Classifier. The main reason for doing this was that the LGBM classifier is known to efficiently handle a large amount of dataset with less memory usage and our dataset contains more than 300,000 lines/rows [1][2]. Moreover, the LGBM classifier is also known to be quite fast and powerful since it uses gradient boosting. Moreover, the gradient boosting used has the advantage of minimizing loss function by using the additive method to add up the weak learners.

Random Forest Model:

The decision to choose Random Forest over other models is because of its properties as an ensemble classifier and its ease of use. Decision Trees alone are of low bias and high variance, but as an ensemble the variance is reduced greatly. This reduces the amount of overfitting while enhancing accuracy. Another big emphasis on random forest rather than Neural Networks is that we can see the variables. It offers insight on understanding the variables, which may be important in understanding the effect of Covid19 . It is also scalable (based on Tree partitions) so there are less parameters to tune compared to SVM, making it easy to use.

Linear SVC Model:

The last model chosen is a linear support vector machine model. The reason for choosing a SVM model is for its ease of use as well as it is very effective on datasets with multiple features. In comparison to other models such as Naive Bayse, SVM is seen to outperform it in most cases. The one downside to an SVM is that it is not suitable for large datasets normally which the processed data is. However, this is why SciKit Learn Linear SVC model was used since it scales better to larger datasets in comparison to a default SMV model. The linear SVC model also contains multiclass support with a one-vs-the-rest scheme which is mandatory for our dataset.

2.3 - Evaluation

For the evaluation of each classification model, we used Sklearn's confusion matrix to extract the true positive (TP), true negative (TN), false positive (FP) and false negative (FN) values for each unique outcome, using our train and validation dataset. Since the model was trained without fine tuning any hyperparameters, there were some zeroes and NaNs because the default model returned zeroes for some or all values of TP, TN, FP or FN. The output of each model is shown below. For this classification problem we believe that accuracy and recall are the most important evaluation metrics. This is because it is important to see how often our classifier is correct, which is accuracy. As for recall we believe it is important since the nature of the dataset is predicting if a patient with COVID-19 is hospitalized, deceased, etc… False negatives in this case are a lot more costly than false positives, which is why we chose recall over precision.

LightGBM Classification Model:

It can be seen in the graph below that precision and recalls are quite low which indicate that false positives and false negatives were a lot more than true positives (as per the formulae). Moreover, the overall accuracy is also not that good which shows that the model did not perform that well on the validation dataset. Overall, this shows that there were a lot of false results as compared to true results since the numbers indicate that FP and FN overpowered TP and TN in terms of quantity.

Deceased	Hospitalized	Non-Hospitalized	Recovered
Accuracy: 0.9878684020835884 Precision: 0.0 Recall: 0.0 F_One: 0.0	Accuracy: 0.6533790308322113 Precision: 0.00347826086956521 Recall: 8.027292795504716e-05 F_One: 0.0001569242840329541	Accuracy: 0.4124335278197125 Precision: 0.408176039391874 Recall: 0.9781992733091103 F_One: 0.5760020413771445	Accuracy: 0.7464332830116828 Precision: 0.06427221172022685 Recall: 0.0038370387089493286 F_One: 0.007241746538871139

Milestone 2 report

Random Forest Model:

The results obtained are subpar. Both deceased and hospitalized labels have NaN and zero values. The model does not predict Hospitalized or Deceased labels for the Testing data, thus we obtain NaN Values. For Non-hospitalized and recovered, Precision=1.0. This means that the False Positives are very low. But the recall, being very close to zero means the false negatives are high, and this is further support in how low the accuracy is.

Deceased	Hospitalized	Non-Hospitalized	Recovered
Accuracy: 0.0 Precision: NaN Recall: 0.0	Accuracy: 0.0 Precision: NaN Recall: 0.0 F_One: NaN	Accuracy: 0.0015039604291300424 Precision: 1.0 Recall: 0.0015039604291300424 F_One: 0.0030034038577053993	Accuracy: 0.0028386510730101055 Precision: 1.0 Recall: 0.0028386510730101055 F_One: 0.00566123188405797

Linear SVC Model:

Again the confusion matrix below tells us the result of our model. From the matrix you can see from the output table that the evaluation metric values for Non-hospitalized are extremely high and hospitalized and Recovered are both subpar or average. As for deceased it has a high accuracy rating at roughly 98%, though the other metrics are either nan or zero. As mentioned above, this is due to our model not predicting any deceased labels correctly making the true positive and false positive all zero. We know this by the formula for precision since it is a nan value. Since f-score relies on both precision and recall we see a zero as well.

Deceased	Hospitalized	Non-Hospitalized	Recovered
Accuracy: 0.9872699824554245 Precision: nan Recall: 0.0 F_One: nan	Accuracy: 0.7180627524582807 Precision: 0.5925366991356839 Recall: 0.5229446664245065 F_One: 0.5555698482119886	Accuracy: 0.9402396398601874 Precision: 0.9124250048383975 Recall: 0.9439698324768071 F_One: 0.9279294055898176	Accuracy: 0.7272566540182518 Precision: 0.44660664149482043 Recall: 0.5169206633796504 F_One: 0.47919804705760144

2.4 - Overfitting

LightGBM Classification Model:

By varying the 'n_estimator' model and plotting the accuracy of the train and validation datasets, we got to see how much the default model overfits for our dataset. It can be clearly noticed in the plot (named 'LGBM_overfitting_graph' inside the plot folder) that the test accuracy with increasing n_estimator parameter was around 41% while the train was around 91%. This huge difference shows us that there was definite overfitting.

Random Forest:

The hyperparameter chosen is the number of Trees in the forest. There is perfect accuracy when running the model on the training data. However, when running on the testing data, our accuracy falls sharply down to 40%. This goes to show that there is very strong overfitting, especially in how the numbers have no impact on the accuracy. The standard parameters set does not do any form of pruning or regularization that is important in managing overfitting.

Linear SVC Model:

The hyperparameter chosen is the C parameter. The reason for choosing the C parameter is that the parameter is looking to minimize error. With the tuning of this parameter there is a bias-variance trade-off. When C is set to a lower value the classifier allows only a small bit of misclassification making the classifier have low bias but high variance due to its generalization to be poor. Converse, if C is large the number of misclassifications allowed are increased. This swaps the values of bias and variance as well. From the C value plots we see that there is a slight gap indicating some variance between the train and test data. However, the gap differs by roughly 3%. This leads us to believe that there is a little bit of overfitting occurring but not enough to make a substantial difference.

Milestone 2 report

References:

[1] https://machinelearningmastery.com/gradient-boosting-with-scikit-learn-xgboost-lightgbm-and-catboost/
[2] https://www.analyticssteps.com/blogs/what-light-gbm-algorithm-how-use-it
[3] https://towardsdatascience.com/3-reasons-to-use-random-forest-over-a-neural-network-comparing-machine-learning-versus-deep-f9d65a154d89
[4] https://scikit-learn.org/stable/tutorial/machine_learning_map/
[5] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[6] https://towardsdatascience.com/support-vector-machines-a-brief-overview-37e018ae310f