Anmol Saini ahsaini@sfu.ca

Jason Cao yjc2@sfu.ca

Jograj Sidhu jograjs@sfu.ca

COVID-19 Project ~ Milestone 2

CMPT 459: Data Mining

1. Additional Preprocessing

Before progressing to the steps of milestone 2, we fixed our *cases_train_processed.csv* file as it was initially only joined on the key which consisted of "province, country," so for rows with *unknown* province values, there were missing values. We updated the file to join on the "country" attribute from *location.csv* in the case where the row could not be joined on key. The "date_confirmation" attribute was also converted into an integer type for easy usability for the models.

2. Tasks

2.1. Splitting Datasets

The *cases_train_processed.csv* file was read in as a dataframe and separated into two sets, one with the outcome column, and the other with all columns excluding outcome. Training and validating any model requires the dataset to be split into train data and validation data. Using sklearn library's *train_test_split* function, the train dataset was partitioned into 80:20 train to validation ratio. The *random_state* parameter was set to 11 to obtain deterministic split results.

2.2. Build Models

The 3 classification models we chose to build are K-Nearest Neighbours (KNN), Random Forests (RF), and LightGBM. The sklearn library was used for all these models, but the API for the boosting tree, LightGBM, was installed separately using *homebrew* on macOS. Since our dataset has categorical features, those must be dealt with appropriately. KNN and RF require categorical attributes to be encoded before being fed to the model for training and predicting, so we used *Label Encoding* to assign each categorical value an integer according to alphabetical ordering. Label encoding is favoured over one-hot encoding which results in a large number of categories(>1000), leading to intractable memory consumption. There is no need for label encoding in LightGBM since it can process categorical features within itself, so the columns with type "object" were converted to type "category." Our reasons for choosing these three models are explained below:

- ***K-Nearest Neighbours*** is a versatile classifier for any kind of classification, and it is also easy to implement. It is a lazy learner which means it does not need training data points to build the model, making the training phase fast. The KNN algorithm has high accuracy and can compete with eager learning algorithms. The KNN algorithm has two main parameters: k and weights. The low number of parameters makes it easier to find the optimal combination of parameters than the other models.
  - K was set to 9, and the weights parameter is the inverse of the 'distance.'
- ***Random Forest*** is a flexible model for classification, as well as regression, therefore it works well with categorical and continuous values. RF uses random sampling to lower the correlation and it consequently minimizes the variance of decision trees, leading to a reduction in overfitting. Since it is a rule-based approach, normalization of the data is not required. RF is also notable for its ability to measure feature importance.
  - The baseline classifier was used, without feature selection and only tuning the *max_depth* parameter. (set to 22).
- ***LightGBM*** is an ideal model choice for multiclass classification. Compared to other boosting algorithms, LightGBM has a very fast training speed. In comparison, it is 7 times faster than XGBoost. Moreover, its use of memory is very low, unlike many other tree-based algorithms. LightGBM is also known for its smart approach to handling large datasets and for its importance on the accuracy of results. With over 100 parameters to tune, it provides highly efficient predictions based on the correct set of parameter tuning.
  - In this milestone, all default parameters were used, with only the exception of *num_leaves* parameters (set to 100).

2.3. Evaluation

The following table summarizes prediction statistics (accuracy, precisions, recalls, f1-scores and support) for both train and validation data. The closer to 1, the higher the classification accuracy. High precision indicates all predictions are virtually correct i.e., the model produces no false positives. High recall indicates actual cases in the dataset are correctly predicted i.e., the model produces no false negatives. F-1 score considers the weighted average of both precision and recall, therefore, taking both false positives and false negatives into account. The support metric describes the number of actual occurrences of the class label in the dataset. Support helps in diagnosing imbalance between the classes, so rebalancing methods can be considered.

Model Prediction Scores

K-Nearest Neighbours

```
TRAINING
Accuracy score: 0.88185
Classification report:
                  precision    recall  f1-score   support

        deceased       0.88      0.33      0.48      3586
    hospitalized       0.81      0.87      0.84    100092
nonhospitalized        1.00      1.00      1.00    119928
       recovered       0.79      0.73      0.76     70494

        accuracy                           0.88    294100
       macro avg       0.87      0.73      0.77    294100
    weighted avg       0.88      0.88      0.88    294100


VALIDATION
Accuracy score: 0.86416
Classification report:
                  precision    recall  f1-score   support

        deceased       0.47      0.11      0.18       913
    hospitalized       0.79      0.86      0.82     24908
nonhospitalized        0.99      0.99      0.99     30062
       recovered       0.76      0.69      0.72     17643

        accuracy                           0.86     73526
       macro avg       0.75      0.66      0.68     73526
    weighted avg       0.86      0.86      0.86     73526
```

Random Forest

```
TRAINING
Accuracy score: 0.87762
Classification report:
                  precision    recall  f1-score   support

        deceased       0.98      0.24      0.39      3586
    hospitalized       0.79      0.88      0.84    100092
nonhospitalized        1.00      1.00      1.00    119928
       recovered       0.79      0.70      0.74     70494

        accuracy                           0.88    294100
       macro avg       0.89      0.70      0.74    294100
    weighted avg       0.88      0.88      0.87    294100


VALIDATION
Accuracy score: 0.86534
Classification report:
                  precision    recall  f1-score   support

        deceased       0.73      0.11      0.19       913
    hospitalized       0.78      0.88      0.83     24908
nonhospitalized        0.99      0.99      0.99     30062
       recovered       0.77      0.67      0.72     17643

        accuracy                           0.87     73526
       macro avg       0.82      0.66      0.68     73526
    weighted avg       0.86      0.87      0.86     73526
```

LightGBM

```
TRAINING
Accuracy score: 0.87861
Classification report:
                  precision    recall  f1-score   support

        deceased       0.93      0.20      0.33      3586
    hospitalized       0.80      0.89      0.84    100092
nonhospitalized        1.00      1.00      1.00    119928
       recovered       0.80      0.70      0.75     70494

        accuracy                           0.88    294100
       macro avg       0.88      0.70      0.73    294100
    weighted avg       0.88      0.88      0.87    294100


VALIDATION
Accuracy score: 0.87062
Classification report:
                  precision    recall  f1-score   support

        deceased       0.66      0.11      0.18       913
    hospitalized       0.79      0.88      0.83     24908
nonhospitalized        0.99      0.99      0.99     30062
       recovered       0.78      0.68      0.73     17643

        accuracy                           0.87     73526
       macro avg       0.80      0.67      0.68     73526
    weighted avg       0.87      0.87      0.87     73526
```

All 3 models are highly accurate in discriminating non-hospitalized cases from other types of cases. This is corroborated by their high F1-scores. The presence of false positives and false negatives (11% to 20+%) appears to render the models less effective in classifying hospitalized and recovered cases. The reduction in F1-scores seems to reflect this observation. Lastly, all 3 models are unable to classify deceased cases given their low F1-scores, which most likely due to low recalls (i.e., high false negatives). In summary, the results of all 3 models follow a general pattern from both their successes and failures. All of them can model the dataset with high degrees of classification accuracy (in the high 80's) by using just the baseline parameters and can classify non-hospitalized cases close to 100%. However, a detailed breakdown of their classification metrics (precision, recall, and F-score) and results from their confusion matrices (knn_cm, rf_cm, and lgbm_cm in the *plots* folder) reveal that all 3 models have difficulties distinguishing between hospitalized and recovered cases. Moreover, all the models classify deceased cases at very low accuracy.

2.4. Overfitting

Overfitting can be identified by checking validation accuracy. While training accuracy can increase indefinitely, validation accuracy usually increases until a point where it stagnates or starts to decline when the model is affected by overfitting. Accuracy curves were constructed for KNN, RF and Light GBM for diagnosing model performance, which can be seen in the *plots* folder labelled as their respective accuracy curves. For KNN, RF, and LightGBM, their accuracies were measured by varying the following parameters respectively: neighbors, max depth and number of leaves.

For RF, the deeper the trees, the more information about the dataset it can capture thus gaining more accuracy. At the same time, noises are also captured along the way, leading to overfit. This was put to the test by varying its max depth. As depicted by the figure *rf_accuracy_curve.png*, separation of accuracies between the test and train data begins at tree depth = 13. The validation accuracy curve stagnates around tree depth = 22, which we chose to set as the parameter value for all the predictions. Another way to control the depth of tree classifiers is to limit the number of nodes. For Light GBM – which is known for leaf-wise tree growth – it is the number of leaves, one of the most important hyperparameters for getting high accuracy and avoiding overfitting. As depicted in *lgbm_accuracy_curve.png*, separation of accuracies between the test and train data begins at num_leaves = 60. The validation accuracy curve that stagnates around num_leaves =100 which is chosen as the optimal parameter for making predictions in this milestone. An important hyper parameter for the KNN algorithm is k number of neighbors. If k is too small, KNN becomes over sensitive to outliers and overfitting may occur. Increasing the value of k helps to smooth out the local outliers but having k that's too large may result in underfitting. From its accuracy curve (*knn_accuracy_curve.png* in the *plots* folder), lower k values provide an initial rise in accuracy scores until k = 40. There's diminished returns at greater values. At k > 50, the accuracy plateaus.

With a maximum of 1% – 2% difference in accuracy between training and validation predictions, all 3 models do not overfit as their chosen parameter varies.