CMPT 459 Fall 2020

Data Mining in COVID-19 Datasets

(Course Project Report)

Presented by**: Team S+**
Dec 10, 2020

George He 301315033
Sam Mao 301388046

Problem Statement

The COVID-19 pandemic has affected a great population across many countries, due to its highly contagious and hazardous nature, lots of research has been conducted around it. Numerous datasets on covid individual cases and location reports from all over the world are made available online. Those public datasets provide an opportunity to start a data mining project, with focus on the analysis of data and generation of accurate models. The models should predict the outcome for individuals infected with covid-19 given specific information as input (location, age, sex etc.). Accurate prediction of outcomes (especially with deceased class) can help determine the severity of individual cases and lead to appropriate treatment.

Dataset Description & EDA

1.0 Dataset Description

Individual Case Dataset:

A .csv file with 557365 records each representing an individual case report.

Attributes: age, sex, province, country, latitude, longitude, date_confirmation, additional_information, source, outcome.

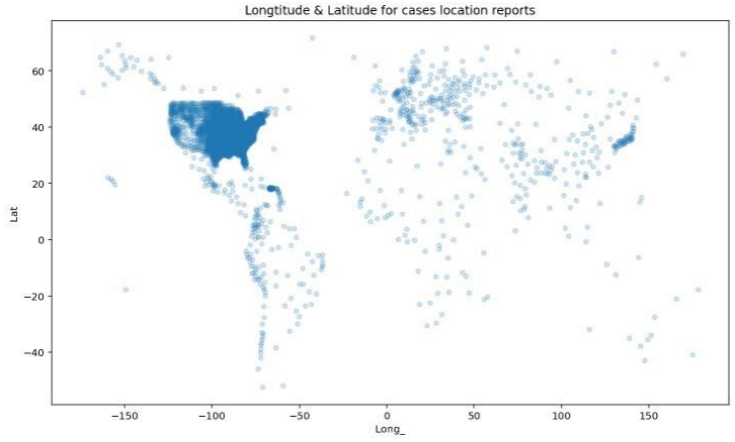
Location Based Dataset:

A .csv file with 3955 aggregated reports from different locations over the world

Attributes: Province_State, Country_Region, Last_Update, Lat, Long_, Confirmed, Deaths, Recovered, Active, Combined_Key, Incidence_Rate, Case-Fatality_Ratio

1.1 Exploratory Data Analysis

Location Based Dataset:



By plotting the latitude and longitude of all location reports, we can visualize places with many reports as darker regions.

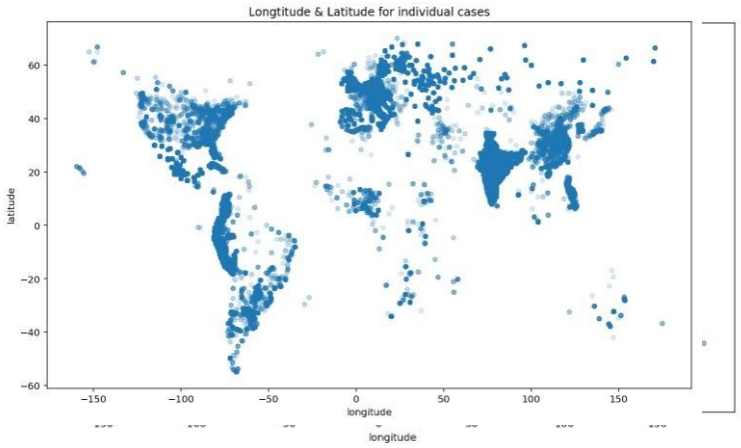
3270 reports came from the US by early June (83% of all reports).

Other discoveries: (refer to plots folder for visualization)

- US, India and Brazil have significantly more confirmed cases than other countries.
- Among all provinces in the US, Texas has the highest number of reports (> 250).
- Most countries have infection rates below 2% and fatality ratio under 10%.

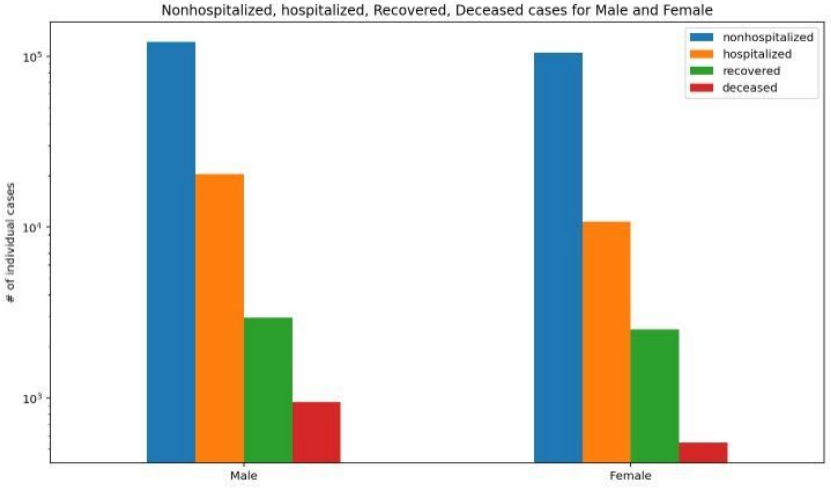
- Almost all reports (3951/3954) are collected on 2020-09-20 04:22:56.
- Yemen has the highest fatality rate near 30%.

Individual Case Dataset:



By plotting all individual cases using latitude and longitude, we can visualize severely afflicted areas as dense and dark areas on the plot.

India is the country with most individual cases by early June (contains more than 50% of total individual cases).



We plotted the outcome count by gender, the bar plot indicates the outcomes between male and female cases follow the same distribution.

Other discoveries: (refer to plots folder for visualization)
- A normal distribution emerged with mode (30, 40] after binning the age attribute.
- The number of new cases increased exponential since January (358049 in May).
- India has most individual cases among all countries in the dataset (301144)
- Aggregated count for each outcome: nonhospitalized (250000), hospitalized (203228), recovered (98137), deceased (5999).

Dataset Preparation

1.2 Data cleaning and Imputing missing values

Individual Case dataset:

More than half of dataset miss values for age and sex attribute. Therefore taking the mean, mode or median to impute for 300,000 records is not a meaningful approach.

- age: All different formats of age are reduced to a single number:
    "x-y" replaced with (x+y)/2
    x months replaced with x/12
    x=nan replaced with -1.
- sex: Missing values are labeled with 'unknown'.
- additional_information: 94% records missing this value, remove this column.
- source: Many missing values, the description of source contains many random websites which are intuitively irrelevant, removed this column from the dataset.
- province, country, latitude, longitude, date_confirmation: Hard to impute for these attributes. Records that miss those values are removed due to small percentage

Location Based Dataset:

- Lat and Long: After examining records that miss these attributes, these records represent reports of ships. Therefore they are removed.
- Fatality Ratio: Some records miss fatality ratio because the number of confirmed cases is 0. These records are removed from the dataset.
- Province and Incidence Rate: Cannot meaningfully impute these attributes. Since the number of missing values (160) is small, they are removed from the dataset.

1.3 Dealing with Outliers

We utilized two methods for detecting potential outliers: z-score and 1.5 IQR

Individual Case dataset:

- latitude and longitude: All latitude and longitude values are within their legal range.
- Age: Records with age > 100 are removed for 2 reasons: 1. they fall outside of 3 standard deviations by z-score and 1.5 IQR. 2. Only 24 records have age > 100.
- Confirmed Date: Every record's confirmed date is within the pandemic period.
- No outliers detected in outcome, province or country attributes.

Location Based Dataset:

- Lat and Long_: All latitude and longitude values are within their legal range.
- Confirmed: Several location reports have exceptionally large numbers of confirmed cases due to the large populations in the cities. Therefore they are kept.
- Deaths, Recovered, Active: Outliers are examined with the conclusion that they all can be explained similar to Confirmed. Therefore they are all kept as valid data.
- Incidence_Rate: Outlier records are found with incidence rate >10%. However, these records can be explained after knowing the cities associated with these records all have small populations (<10000). Therefore these data are valid.
- Case-Fatality_Ratio: Outliers were found with value > 15%. The high fatality ratio can either be explained by the small number of confirmed cases (7 confirmed and 2 deaths will

yield a fatality ratio of 28%), or the country of origin (Mexico has a really high fatality ratio). Therefore those records are kept as valid data.
- Arithmetic Check: Deaths + Recovered + Active = Confirmed for all records

1.4 Transformation & Join two datasets

In order to join both dataset using common attributes, an aggregate method is identified for each of the following attributes in location dataset:
- Lat and Long_: Since the individual case dataset also has latitude and longitude, the method for these two attributes is trivial and columns can be removed.
- Confirmed, Deaths, Recovered, Active: We took the sum as representative for each of those attributes during aggregation.
- Incidence_Rate: Since the city population is lacking from the dataset, the average of incidence rate among all cities serves as the best estimate for that province.
- Fatality_Ratio: The fatality ratio for each province is recalculated by dividing deceased by confirmed cases after joining the two datasets.

1.5 Methods of Joining

We chose country and province as attributes for joining both datasets. Because joining by latitude and longitude will produce mismatched records for two reasons:

1. In order to match provinces with individual cases based on location, we need to set a maximum offset which is the difference acceptable between two matched records in terms of latitude and longitude. The selection of such an offset is difficult as values too large or small will not work (small values will leave many records unmatched, large values will mismatch many records).

2. Provinces that are small and close together will yield location coordinates that are very similar, it is easy to mismatch records close to these provinces.

Thus we chose to merge two datasets by country and province as we believe this is the most accurate and safe approach. The joined table yields about 450,000 records.

Classification Models

2.1 Encoding & Splitting Dataset

Through experiment, the employment of one hot encoding on 4 categorical feature columns significantly increases the dataframe size and training time without improving classification accuracy of any chosen model. Thus, label encoding is chosen to convert all five categorical columns into numeric columns. To split the dataset into training and test sets, train_test_split from sklearn is used with test_size = 0.2.

2.2 Choice of Models

After examining the performance of several models with default parameters (LightGBM, Adaboost, XGBoost, SVM, NaiveBayes, KNN), we selected LightGBM and KNN as models to train, optimize and evaluate in milestone 2 and 3.

When using default hyperparameters, LightGBM has shorter training time, with same classification accuracy when compared to both XGBoost and Adaboost. LightGBM is based on decision tree algorithms, yet its leaf wise split approach produces trees with complex structure which can lead to higher accuracy. Since there are many hyperparameters associated with boosting trees, choosing LightGBM will make the tuning phase more efficient while achieving higher accuracy.

KNN has moderate training time and high accuracy score(around 0.87) with k = 10. KNN classifier is robust to skewed distribution of data and does not require classes to be linearly separable. Compared to other models, KNN also has less hyperparameters to tune. Though a meaningful distance function is desired along with feature reduction and normalization to achieve best performance, these hyperparameters can be tuned in milestone 3. Thus, we have selected KNN as our second model.

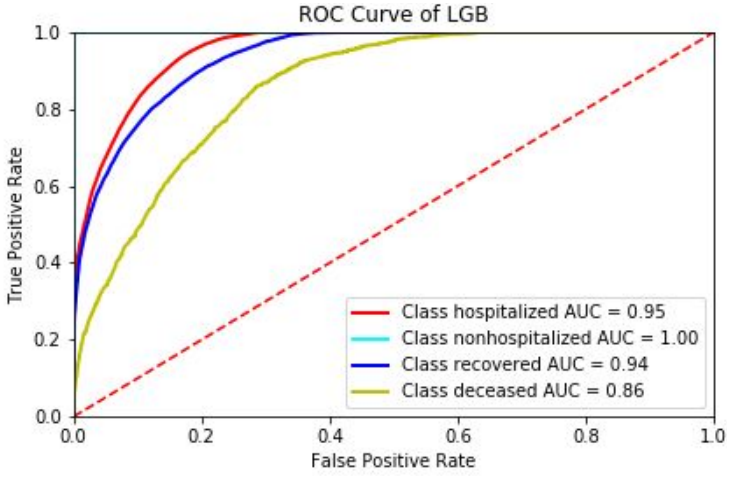
Initial Evaluation & Overfitting

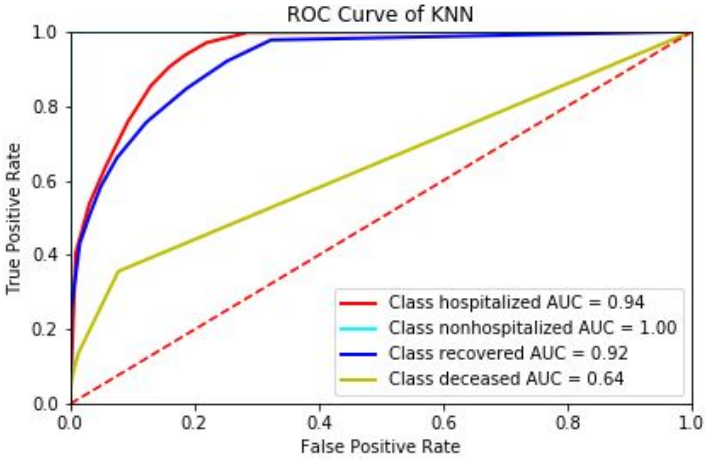
2.3 Evaluation of Models with Default Parameter

	Metrics	Accuracy Score	Precision	Recall	F1 hospitalized	F1 nonhospitalized	F1 recovered	F1 deceased
LGB	Train	0.879	0.881	0.879	0.873	1	0.678	0.169
	Test	0.873	0.871	0.874	0.867	1	0.667	0.078
KNN	Train	0.859	0.857	0.859	0.854	1	0.614	0.084
	Test	0.855	0.852	0.855	0.849	1	0.606	0.071

When evaluating model performance on an imbalanced dataset, accuracy scores cannot serve as an effective measure. If the class distribution is skewed with little training examples on the minority class (deceased in our case), the overall accuracy score will mainly be affected by predictions made on the other majority classes. Therefore, F1 score is a better metric as it also measures the incorrectly labeled cases (both false-positives and false-negatives cases). However, F1 scores are harder to interpret.

Below are the ROC Curve plots for both models on test dataset:





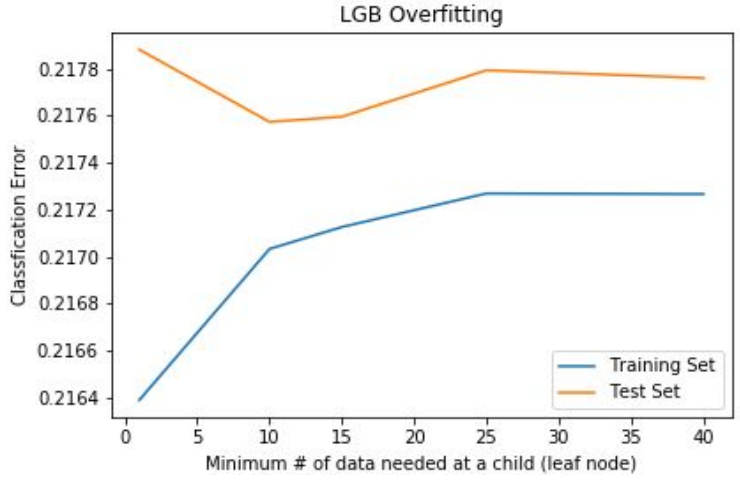
Accuracy score captures only the precision, and F1 score captures only one of the possible trade-offs between precision and recall. The ROC curve is the best performance measure of the models as it visualizes the trade-off between TP and FP under different thresholds (0-1). The ROC curves are independent from the class distribution, making them appropriate for evaluating models on skewed data with visual comparison of different classifiers.

Classifiers that have better performance will produce curves closer to the top-left corner. The closer the curve to the red dotted diagonal line, the less accurate the predictions made by the model to the corresponding class.

The top 3 curves on both plots are almost identical. This means both models can accurately predict data with class labels hospitalized and recovered, with almost perfect predictions for data with label non hospitalized (AUC = 1). However, the performance on data with deceased labels is much poorer with LGB (AUC = 0.86) outperforming KNN (AUC = 0.64).

2.4 Overfitting

We have observed overfitting for LightGBM. First Fix the number of estimators to 1 (1 tree in the classifier), and the maximum number of leaf nodes in that tree to 1000. Then train 5 different models with different values for min_child_samples (minimum number of data required at a leaf node). The following plot shows the classification error of trained models.



is

but

When the only tree in each classifier allowed to grow as large and as deep as possible (with more pure leaf nodes), the model starts to overfit. The overfitting trend starts when min_child_samples < 10, where the classification error of the model decreases on training data increases on test data. (lgb_overfitting.png)

We did not observe overfitting in KNN models. By training models for different k_neighbors and plotting the classification errors, the error of both the training set and test set stays very close and has the same trend for all models. For plot please refer to (knn_overfitting.png)

Hyperparameter Tuning

The dataset was splitted into training dataset and test dataset with ratio 75:25

3.1.1 Tuning LightGBM

After reading the documentation of all parameters and conducting a few experiments, we classified all important parameters in LGB model into three categories:

1. Categorical parameters that will affect the types of base model chosen or approaches taken by the classifier:
    a. **boosting_type:** 'gbdt', 'dart', 'goss', 'rf'
    b. **objective:** 'regression', 'binary', 'multiclass', 'lambdarank'
    c. **class_weight:** None, 'balanced'
2. Important numerical hyperparameters that will affect model performance:
    a. **num_leaves:** Maximum tree leaves for base learners.
    b. **min_child_samples:** Minimum number of data needed in a child (leaf).
    c. **max_depth:** Maximum tree depth for base learners.
3. Less important numerical parameters to further improve model performance:
    a. **learning_rate:** Boosting learning rate.
    b. **subsample_for_bin:** Number of samples for constructing bins.
    c. **n_estimators:** Number of boosted trees to fit.

Method

GridSearchCV was selected as the exhaustive search method for tuning LightGBM.
- The number of cross validation folds was set to 5 (default in **GridSearchCV**).
- Two evaluation metrics were used in the scoring function: the recall of minority class (deceased) and the overall unweighted mean of all classes' recall.
- Refit scoring function was set to each of the evaluation metrics. The model that achieves the best recall on deceased class will be used for 3.2 comparative study.

Techniques

- Parameters in category 1 are easy to tune, due to the limited choices of values. The **objective** was set to 'multiclass' as we are dealing with a multiclass classification problem.
- After finding the best values for parameters in category 1, we fix those parameters and continue gridsearch on parameters from category 2.
- After fixing values for parameters in category 1 and 2, gridsearch on parameters from category 3 was executed last, in the hope of further increasing model performance.

3.1.2 Tuning KNN

We chose 3 parameters to tune for the KNN model..

1. **n_neighbors: int**, number of nearest neighbors
2. **weights : {'uniform', 'distance'}**
    - 'uniform' : points in each neighborhood are weighted equally.
    - 'distance' : weight points by the inverse of their distance.
3. **p : int**, power parameter for the Minkowski metric. p=1, manhattan distance(L1), p=2 euclidean distance(L2).

GridSearchCV was used for tuning with three evaluation metrics, i.e. Accuracy, Overall Recall, Recall on 'deceased'. Cross validation folds was set to 5. We define hyper-parameter space for searching as follows:

hyper-parameter space:

- **n_neighbors**: [5, 15, 25, 35]
- **weights**: ['uniform', 'distance']
- **p**: [1, 2]

Results

LightGBM

Using overall recall as the evaluation metric will produce a model that yields the highest recall score on deceased class. The best parameter settings are shown in the table below.

KNN

Using accuracy or overall recall as evaluation metrics is better, as they will produce models that yield the same recall but higher precision on the deceased class.

Result on Training Dataset:

Model	Hyperparameters	Accuracy	Overall Recall	Overall Precision	Recall on 'deceased'	Precision on 'deceased'
LightGBM	boosting_type = 'goss' objective = 'multiclass' class_weight = 'balanced' num_leaves = 200 max_depth = -1 (no limit)	0.75	0.76	0.66	0.80	0.05
KNN	n_neighbors = 25 weights = 'distance' p = 1	0.88	0.66	0.90	0.10	0.99

We noticed that LightGBM achieves higher recall on the deceased class (0.8), by sacrificing many correct predictions for other classes (precision = 0.05). This is due to the metric chosen and class weights specified in GridSearchCV. The parameters and refit metric will force the classifier to favor the minority class.

KNN captures much less deceased cases (recall = 0.1), but almost all captured cases are true-positive (precision = 0.99).

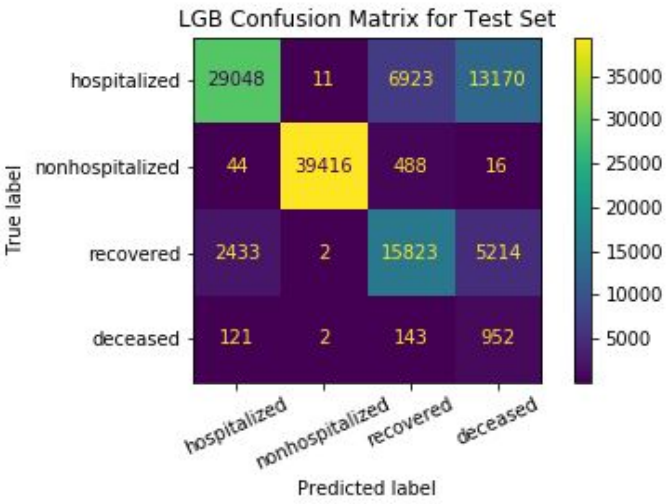
Conclusion

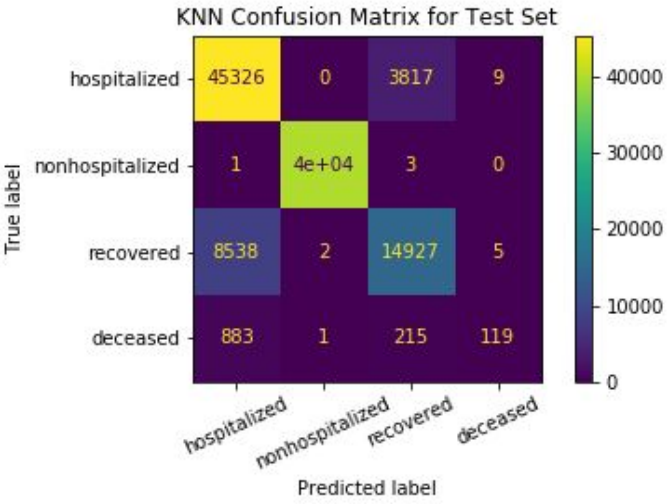
Result on Test Dataset using best models:

Model	Accuracy	Overall Recall	Overall Precision	Recall on 'deceased'	Precision on 'deceased'	F1 on 'deceased'
LightGBM	0.75	0.76	0.66	0.78	0.05	0.09
KNN	0.88	0.66	0.88	0.1	0.89	0.18

The performance of LightGBM on the test dataset is similar to the training dataset, with high recall (0.78) and low precision (0.05).

KNN still identifies 10% of all deceased cases, but precision dropped to 0.89.





The confusion matrices above show that LightGBM captures more deceased cases and has higher recall (0.78) while KNN has higher precision (89%).

Since the goal is to have less false-positives for deceased class, the LightGBM is the best model for achieving high recall.

Lessons Learnt & Future Work

Lessons Learnt:

- The tradeoff between precision and recall was reflected in both models.
- Accuracy may not reflect the true model performance when data is imbalanced.
- Prediction for minority class can be hard as there are few examples to learn from.

Future Work:

- A self-written distance metric for KNN models may further improve its performance.
- Majority of deceased cases were mis-classified as hospitalized. It is important to learn the key features that are responsible for the difference between these two classes.
- The F1 score on deceased class could be used as the metric for tuning models.
- A randomized search could be performed on the narrowed hyper-parameter space, to improve the optimal parameters.
- Apply learning based CNN models on this dataset, by comparing our results to learning based methods, we could have a better view about how much learning based techniques can improve on the classification tasks.

Contribution

George: Data cleaning & EDA on individual case dataset, Transformation & Joining datasets. Build, evaluation and overfitting in LightGBM model. Parameter tuning in LightGBM & comparative study.

Sam: Data cleaning & EDA on location based dataset. Build, evaluation and overfitting in KNN model. Hyperparameter tuning on KNN & comparative study.