Effective Methods for Capturing Cattle Rustlers

Yung-Hsin Chen[1] and Haoxin Cai[1]

[1] Universität Zürich, Zürich, Switzerland

19 December 2022

The goal of the report is to get the most related factors of COVID-19 pandemic cases in countries. In this report, data are collected from 192 countries, and classification models are applied in order to get a list of feature importance. It is believed that the more-important-features play more crucial role in the severity of the pandemic of a certain country. With an accuracy of 74.36%, XGBoost model suggests that human development index, life expectancy, population density, hospital beds per thousand and GDP per capita of a country are the leading factors of the severity of the pandemic.

1 Introduction

The COVID-19 pandemic has severe impacts on almost every aspect around the world. It causes not only social and economic disruption, but also drastic death rates. Inevitably, people are curious about the causalities of COVID-19 and how to prevent the pandemic from getting worse. Therefore, the report aims to determine the correlation between the severity of the pandemic other information of a country. Notes that the time parameters are taken out of consideration for simplicity.

2 Domain Knowledge

In this section, tools and domain knowledges used in the task and reasons of choosing them will be briefly explained including models used (logistic regression, linear perceptron, XGBoost), evaluation metrices (micro-f1, macro-f1) and visualisations (confusion matrix, normalised confusion matrix).

2.1 Models

The models used in the task are linear perceptron, logistic regression and XGBoost. The three models are chosen to adapt the small data size of the COVID data. All models have limited parameters and are thus less likely to overfitting, which can happen easily with small datasets. The linear perceptron is the simpliest model and will serve as the base-line model.

2.1.1 Linear Perceptron

Linear Perceptron is a very simple model for binary classification, which can later be adapted to multiclass classification repeating the process with paired up the classes. The architecture of linear perceptron is straightforward. After each data are multiplied by weights and biases, the result (a scalars) will be passed to the activation function. In the case of unit-step function as the activation function, numbers larger than 0 will be assigned to a class, and those smaller than 0 will be assigned to the other class. The whole process can be summarised by Equation 1.

$$y_{\text{predicted}} = \text{unit-step}\left(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots\right) \quad (1)$$

The output with a list of predicted classes will be compared with the actual classes by a loss function. The weights and biases will then be updated by applying stochastic gradient dcecent to the loss function. The perceptron is gauranteed to converge if the classes is linearly separable, but has the problem of stucking in the local minima.

Although linear perceptron is fast to implement and is less inclined to overfitting with small datasets, Equation 1 gives away the linearity decision surface nature of logistic regression. It also indicates the no-multicollinearity requirements between independent variables and the high sensitivity to outliers like linear regression does. Besides, it only takes on yes and no outputs and does not consider calibrated prababilities.

2.1.2 Logistic Regression

Logistic regression is a linear model commonly used for classification problems as well. It is very similar to the linear perceptron model except for the activation function. The activation function for logistic regression is Sigmoid function. For logistic regression, regularisation terms are also added to the loss function as a penalty for overfitting.

With the Sigmoid function in mind, it is obvious that logistic regression is mostly used for binary classification and can be extended to multi-class classification like linear perceptron.

Similar to linear perceptron, logistic regression is also sensitive to outliers and does not allow multicollinearity. However, it no longer takes on only yes or no as outputs because the real world is not likely a binary answer world. Instead, it allows continuous probabilities for outputs, which is fitted by the Sigmoid function. With probability over 0.5, the output is assigned to a class, and with probability lower than 0.5, the output will be assigned to the other class. The regularisation terms also prevent the model from overfitting.

2.1.3 XGBoost

XGBoost is a gradient boosting model with extra structures that has been proved well performed on classification tasks. The boosting model creates weak learner (decision tree) one by one. The previous weak learner set a larger weight for the wrong answers as the input to the next weak learner. By going through the sequence, the classification will eventually be finalised correctly. Each weak learner will have their own predictions, and together they will have to vote to decide what the final answer is. The less accurate weak learners get less votes.

XGBoost has the advantage of tackling non-linear problems and does not require non-multicollinearity on data. It is also very efficient due to the parellel tree boosting and less parameters to be trained comparing to deep neural networks. Although the boosting models are greedy, resulting in higher chance of ovrfitting, setting the correct maximum depth of the trees can effectively avoid this problem.

2.2 Evaluation

Since it is a classification task, f1-score is commonly used for model evaluation. However, f1-score is only calculated per class. The multi-class case will require the aggregation of f1-scores of each class to determine the overal performance of the models on all classes. Micro-f1 and macro-f1 are two different ways of aggregating the f1-scores of each class and will be used as the evaluation metrics of this task. In addition, the confusion matrix and the normalised confusion matrix will be used as the visualisation of the evaluation.

2.2.1 Metrics

After training, the model will be tested on the testing dataset. The performance of the models will then be tested by the evaluation metrics. Metrics used for this task are micro-f1 and macro-f1.

F1 score is a metric of taking precision[1] and recall[2] into into consideration at the same time per class. F1-score is defined as Equation 2.

$$\begin{aligned} \text{f1-score} &= 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \\ &= \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP+FN})} \end{aligned} \tag{2}$$

Since f1-score is calculated per class, to calculate the aggregation of multi-class will become tricky. This is where micro-f1 and macro-f1 come into play. They are two different ways of aggregating multi-class f1-scores. Macro-f1 calculates the average of f1-scores of all classes as Equation 3. The equation shows that macro-f1 gives same weights to each class no matter how large the class is.

$$\text{macro-f1} = \frac{\text{sum(f1-score)}}{\text{number of classes}} \tag{3}$$

On the other hand, micro-f1 is defined by Equation 4. This equation is the same as the one for f1-score (Equation 2). However, the TP, FP and FN stands for the sum of the metrices for all classes.

$$\text{micro-f1} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP+FN})} \tag{4}$$

This shows that f1-score views each observation points equally important. This might cause a bias of measurement with imbalanced datasets. With f1-score, larger classes with more observation points will have a larger impact on the f1-score. A comparison and discussion of the two metrices will be explained more in section 5.

2.2.2 Visualisation

To visualise how well the model performs, a confusion matrix will be the most suitable tool. The confusion matrix aims to check if the model is classifying correctly

[1] According to Layman definition, percision means, of all the positive predictions I made, how many of them are truly positive?

[2] According to Layman definition, percision means, of all the actual positive examples out there, how many of them did I correctly predict to be positive?

Table 1: *Data Summary Table*

Information	Description
number of columns	67
number of rows	236386
number of countries	248
key	{location:date}
starting date	01.Jan.2020

Table 2: *Data Summary Table*

Selected Attributes	
aged_65_older	cardiovasc_death_rate
aged_70_older	diabetes_prevalence
gdp_per_capita	hospital_beds_per_thousand
population_density	human_development_index
life_expectancy	median_age
people_fully_vaccinated_per_hundred	

or confusing different classes. Each number in the row represents the number of instances of the actual class, while each number in the column is the number of instances of the predicted class. The brighter the squares indicates the more number of instances. Ideally, the diagonal squares should be the brightest. However, some class has fewer data, resulting in darker colour of squares even the data are all classified correctly. Therefore, in the report, both confusion matrix and normalised confusion matrix will be shown. The normalised confusion matrix helps elimiate the unbalanced dataset issue.

3 Method

In this section, the methods of this particular task will be explained, including introduction of data, building features for the models, training and predicting classifiers and generating the feature importance table. Classifiers and other used tools are introduced and explained in section 2.

3.1 Data

Data used for this task is from covid-19-data from Our World of Data. It is online in a github repository, making it easy to access. The data is loaded into the local database by the *request* package of Python. Each time the data is downloaded, the newest data (up-to-date) will be fetched from the github repository. Thus, the result might be slightly different every time. The result of this report is executed with data up til 30. Nov. 2022.

The data consists of 67 columns and 248 countries with 236386 rows in total. Attribute *location* and *date* together define each unique row of data. The data is still being updated. A summary table of the data is shown in Table 1.

3.2 Building Features

The process of building features includes data cleaning and feature selection, and label preparation. Data cleaning and feature selecting is crucial for the accuracy of models. Bad data cleaning can lead to biased results or bad model performances. Relevant attributes will be selected as features to be put into the classifiers, i.e., the models. Finally, since this is a supervised learning task, the label should be prepared for model training.

In the data cleaning phase, the goal is to get a table of one row per country, i.e., each row represents the information of a country. The label is defined as the attribute, *total_cases_per_million*. To achieve this, relavent attributes are first selected for model training. For this task, eleven attributes that is speculated to affect the number of COVID cases are selected from the raw data. The selected attributes are listed in Table 2. Countries with less than 200 rows of data is then removed. Since each row corresponds to one date, it is not ideal to use the data of a country if less than 200 days of data are recorded. Among the selected attributes, *aged_65_older* and *aged_70_older* are divided by *population* into *aged_65_older_percentage* and *aged_70_older_percentage* respectively so that the models are trained on the percentage of elder people instead of the total number. Except for the attribute *people_fully_vaccinated_per_hundred*, all the other attributes have a single value throughout the dates for each country. However, the attribute *people_fully_vaccinated_per_hundred* and the label attribute *total_cases_per_million* is accumulated day by day. In this case, the data of the latest date is used as the feature value for each country. By doing this, the model will be able to generate the feature importance table according to how all features affect the number of total cases per million in each country. After data cleaning and feature selection, 194 countries/rows and 11 features are left for model training.

After data cleaning and feature selection, label preparation is performed. The attribute *total_cases_per_million* is categorised into four levels of severity. The interval of the categorisation is shown in Table 3. Apparently, there is no country with over 700'000 cases per million. [0, 50000, 200000, 400000, 700000]

3.3 Classifiers

Classifiers are used for classification tasks. Due to the small dimension of the dataset, several changes are made to adapt the data size. The data is not splitted

Table 3: *Label Interval for Label Preparation*

Level	Interval
0	0 - 50’000
1	50’000 - 200’000
2	200’000 - 400’000
3	400’000 - 700’000

Table 4: *Model Summary*

Model	Details
Logistic Regression	
penalty	l2
solver	newton-cg
Linear Perceptron	
tolerance	0.001
random state	0
XGBoost	
learning rate	0.1
loss	deviance
max depth	3
n_estimators	100
random state	21

into training, validation and testing datasets, but only training and testing only. In this task, the training-testing split will be 80% and 20% respectively. Besides, it is not recommended to use models with too many parameters since it might have to higher chance of overfitting. Thus, simple models are picked out for this task including logistic regression, linear perceptron and XGBoost. These models can be easily applied to the dataset with *sklearn* from Python. For hyperparameter selection, grid search cross validation is used. A summary of models used for the task and the best performing hyperparameters chosen by applying grid search cross validation are listed out in Table 4.

3.4 Feature Importance Table

The feature importance is generated automatically via the trained model. It shows weight of each feature. The weight can be thought of as how significant each feature effects the classification accuracy. The more positive the feature importance score, the more the feature helps reduce the loss while training. However, if the feature importance is negative, the feature increases the loss while training.

4 Results

By applying the test dataset on the model, the performance of the models can be evaluated. The accuracy summary of each model and the feature importance table from the best performing model is shown in Table 5 and Table 6[3]. Among the models, XGBoost has the best performance. Among the features, the model suggests that human development index[4], life expectancy[5], population density, hospital beds per thousand and gdp per capita are the top five items that affect the covid cases per milliom in a country.

Table 5: *Model Accuracy Summary*

Model	Accuracy	Micro-f1	Macro-f1
Logistic Regression	64.10%	0.6410	0.5863
Linear Perceptron	48.72%	0.4872	0.2976
XGBoost	76.92%	0.7692	0.7749

Table 6: *Feature Importance from XGBoost*

	Feature	Importance
1	human_development_index	0.462551
2	life_expectancy	0.462551
3	population_density	0.138737
4	hospital_beds_per_thousand	0.082238
5	gdp_per_capita	0.065485
6	people_fully_vaccinated_per_hundred	0.059118
7	cardiovasc_death_rate	0.045620
8	aged_70_older_percentage	0.033860
9	median_age	0.029610
12	diabetes_prevalence	0.026468
11	aged_65_older_percentage	0.025800

5 Dicussion & Conclusion

From the feature importance table, it can be concluded that the severity of covid is more related to the human development index. The human development index is a score on the quality of living in the country including average life expectanvy, GDP, education opportunity, etc. The number of people fully vaccinated, the population composition or the diabetes prevalence do not play the most important role in the total cases as expected.

[3]The result can be replicated by taking the data up to 30. Nov. 2022.

[4]According to Our World In Data, human development index is: A composite index measuring average achievement in three basic dimensions of human development—a long and healthy life, knowledge and a decent standard of living. Values for 2019, imported from http://hdr.undp.org/en/indicators/137506

[5]According to Our World In Data, Life expectancy is: Life expectancy at birth in 2019