Models for predicting COVID19 case severity

Kieran Lankester

I. Introduction

After an initial outbreak in 2019, the COVID-19 pandemic has been ongoing throughout the world. As of the end of March 2021, there had been over 120 million confirmed cases of the virus worldwide, with the death toll passing 2.8 million [1]. With vaccinations and lockdowns being introduced across the world, it is hoped that spread of the virus will gradually decrease and eventually lead to some aspect of containment.

The extent of the pandemic has produced large amounts of data relating to confirmed cases. This has allowed for many machine learning implementations aiming to monitor and anticipate the spread of the virus. In particular, this project uses a dataset of individual-level data collected from national, provincial, municipal, and online health reports [2][3].

The aim of this project is to be able to predict the severity of a case based on specific factors. Features chosen to train the model were age, sex, country of origin, chronic disease status, travel status, and the length of time between symptoms and confirmation. These were used to predict the outcome feature, which is being used as the label.

The dataset consists of roughly 2.7 million samples and 32 features, both numerical and categorical. Therefore, the data would need to be adequately prepared before compatibility with a machine learning model is achieved. Three machine learning classification algorithms will be trained using the prepared data. For each one, different hyperparameters will be tested to attempt to produce the best outcome.

Python 3.8 was used to implement and analyse the models, along with various methods from the scikit-learn library. The Pandas library was used to store and manipulate the dataset.

II. Data preparation

There were a lot of missing values in the original dataset. Any samples that had missing values relating to the outcome or any of the features used, except for date of onset symptoms, were dropped from the dataset. This produced around 34,000 remaining samples. Due to the size and nature of the dataset, there was still a lot of preparation needed for each feature and the outcome before the dataset was ready to be used to train the models. The conclusion of the data preparation stage resulted in a complete dataset with 33,320 samples.

A. Age

Ages were recorded as strings, with some stored as ranges rather than specific numerical values. To remedy this, the strings were split on the occurrence of a hyphen and converted into a list, this was to account for the entries that included ranges. A function was applied to each entry that converted each item in the list to a float and returned the mean value.

B. Sex

The entries for sex just consisted of two strings, male and female, so all that was needed was to convert these to corresponding numeric values. The values used were 0 for male, and 1 for female.

C. Country

Categorical encoding was needed to transform the country feature into numeric values. As the countries were not ordinal, the approach used was One-Hot Encoding [4], which creates additional features based on the number of unique values the original feature includes. There was a large sample size of countries, putting this method at risk of multicollinearity. Variance Inflation Factor (VIF) [5] was used to assess whether or not this issue would arise. Most of the scores for each individual country were close to 1, as seen in Table I, which indicates a much lesser chance of multicollinearity, with only one feature having a score of 5, which due to the size of the dataset shouldn't present too much of an issue.

For comparison, the countries were grouped by continent using the library pycountry_convert and encoded in the same way. This produced similar results, as seen in Table II. Removing the country / continent with the highest VIF score was also tested, but this increased the VIF scores of other features such as age. The values produced when removing a country / continent can be seen in Table III and Table IV, respectively. In the end, the countries were used as feature as this led to higher accuracy when training the models.

TABLE I. VIF SCORES FOR COUNTRIES

Feature	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	c17	c18	c19	c20
VIF	1.00	1.00	1.03	1.00	1.00	1.00	1.01	1.00	1.00	1.00	1.12	1.00	1.01	1.05	1.01	1.01	1.01	1.00	1.01	5.47
Feature	c21	c22	c23	c24	c25	c26	c27	c28	c29	c30	c31	c32	c33	c34	c35	age	sex	cdb	thb	
VIF	1.02	1.01	1.01	1.00	1.01	2.18	1.00	1.15	1.03	1.00	1.00	1.01	1.01	1.04	1.02	1.05	1.01	1.16	1.60	

TABLE III. VIF SCORES WITH COUNTRY REMOVED

Feature	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	c17	c18	c19
VIF	1.00	1.00	1.03	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.11	1.00	1.01	1.04	1.01	1.01	1.01	1.00	1.01
Feature	c20	c21	c22	c23	c24	c25	c26	c27	c28	c29	c30	c31	c32	c33	c34	age	sex	cdb	thb
VIF	3.09	1.02	1.01	1.01	1.00	1.01	1.00	1.14	1.03	1.00	1.00	1.01	1.01	1.03	1.02	3.23	1.47	1.16	1.64

TABLE II. VIF SCORES FOR CONTINENTS

Feature	c1	c2	c3	c4	c5	c6	age	sex	cdb	thb
VIF	1.10	6.34	1.00	1.01	1.02	1.00	1.01	1.00	1.11	1.18

TABLE IV. VIF SCORES WITH CONTINENT REMOVED

Feature	c1	c2	c3	c4	c5	age	sex	cdb	thb
VIF	1.07	1.00	1.00	1.02	1.00	1.46	1.42	1.11	1.21

D. Chronic disease status

Chronic disease status only contained Boolean values, True and False. These were converted to numerical values, 1 for True and 0 for False.

E. Travel status

Travel status also consisted of just Boolean values, True and False. These were prepared in the same way as for chronic disease status.

F. Confirmation date

Confirmation date contained strings as values which represented a date in the format DD/MM/YYYY. This was converted to a date object using the datetime library. A date object representing 01/01/2020 was then subtracted from this to give an integer value representing the number of days from this date to the confirmation date.

G. Onset symptom date

Onset symptom date was converted to an integer representation of days using the same method as for confirmation date, however, around 90% of the samples had a missing value for this feature. As this was the only feature with missing values, iterative imputation [6] was used to fill these missing values. This is a way of estimating the missing values based on the other features and works much like a multiple imputation by chained equations (MICE) algorithm. This was done using the sklearn method IterativeImputer, with maximum iterations equal to 10, and verbose equal to 0.

H. Length of time between symptoms and confirmation

A new feature, of the length of time between symptoms and confirmation, was then created by subtracting the integer representing onset symptom date from the one representing the confirmation date. Following this the two latter mentioned features were removed.

The imputation used on the missing values for onset symptom date caused some outliers to appear in this new feature. The range for acceptable values was set to [-21,21] to account for this. This was set in line with the average amount of time (3 weeks) that symptoms can persist. The distribution of values after making this change can be seen in Fig. 1.

I. Outcome

The label, or the outcome, needed to be transformed to be representative of the two possibilities set out in the initial problem. A list of keywords was produced that deduced whether the specified outcome indicated recovery, 0, or otherwise a more severe case, 1. The most common outcome among the samples was ‘hospitalized’, it was decided that this would be included in the more severe category.

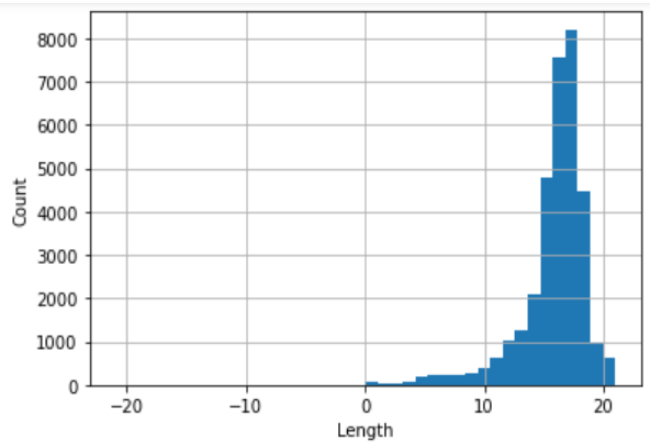

Fig. 1. Distribution of symptoms to confirmation date.

III. TRAINING MODELS

The three machine learning algorithms chosen were Random Forest Classifier, K Neighbors Classifier, and Support Vector Classifier. Each of these used the same training and testing sets, which were generated using a ratio of 75:25 on the respective dataset. The performance metrics used to evaluate the models were the accuracy score, precision score, recall score and area under the receiver operating characteristic (ROC-AUC) value. For each model, different hyperparameter values were tested in an attempt to achieve the configuration resulting in the optimal outcome for the given dataset. The features were scaled using normalization and standardization and the accuracy result of this was compared with the unscaled results. Stratified K-fold Cross-validation was implemented using a k value of 5 to estimate the robustness of the models on unseen data.

A. Random Forest Classifier

The random forest (RF) classifier creates a set of decision trees from randomly selected subsets of the training set. It then aggregates the votes from different decision trees to determine the final class of the test object [7]. It helps to improve accuracy by reducing overfitting in the decision tress, and also works well with categorical and continuous values. Although, the combination of decision trees can require more time for training in relation to other classification models.

Training the random forest classifier using the given training and testing sets with default hyperparameters resulted in an accuracy score of 96.7%, a precision score of 91.2%, a recall score of 94.1%, and an ROC-AUC value of 0.996.

The hyperparameters chosen to optimise for random forest classification were n_estimators, the number of trees in the forest, max_features, the number of features to consider when looking for the best split, max_depth, the maximum depth of the tree, and criterion, a measure of the quality of the split. A slightly higher number of trees than the default of 100 produced better results than a lower number and provided less computational time than a much higher number, for this reason 200 was chosen for the n_estimators parameter. The default value of auto was chosen for the max_features parameter as changing this did not have a noticeable effect on

the outcome. 8 was chosen for the max_depth parameter, increasing the depth from the default of 0 slightly increased the performance of the model, this increase gradually became less noticeable and it was decided to be insignificant once 8 was reached. Gini impurity was chosen for the criterion parameter as this performed better than the information gain of entropy.

Retraining the model with these hyperparameters resulted in an accuracy score of 96.8%, a precision score of 90.3%, a recall score of 96.9%, and an ROC-AUC value of 0.993. Comparing this with the default model, there was a slight increase in accuracy, along with a significant increase in recall. The ROC-AUC value and precision score dropped slightly, however, there seems to be a slight improvement in the model overall.

Applying standardization and normalization to the features prior to training the model provided no change in the results or accuracy and therefore is not significant for this model.

Using the stratified k-fold cross-validation approach with random forest classification resulted in a mean accuracy score of 96.6%, indicating the model will still perform well on average with unseen data.

A comparison of evaluation metrics throughout the random forest model configuration can be seen in Table V, Fig. 2 shows the corresponding confusion matrices. As the standardization and normalization results were equivalent to the base results, these confusion matrices were not included.

B. K Neighbors Classifier

The K neighbors (KN) classifier uses a k value and distance metric to measure the distance of new points to nearest neighbors [8]. The algorithm is simple and intuitive, is constantly evolving and does not explicitly build a model, instead tagging new data entries based on historical data. Although, it does not perform well with imbalanced data and has no capability for dealing with missing data, the latter of which was accommodated in the data preparation step.

Training the random forest classifier using the given training and testing sets with default hyperparameters resulted in an accuracy score of 96.6%, a precision score of 91.2%, a recall score of 93.6%, and an ROC-AUC value of 0.992.

The hyperparameters chosen to optimise for K neighbors classification were n_neighbors, the number of neighbors to be used, and weights, the weight function used in the prediction. The default value for n_neighbors is 5, and the value chosen for the model was 8. It was found that increasing this value much didn't provide an overall improvement for the model. A uniform weight function was chosen for weight, meaning that all points in each neighborhood are weighted

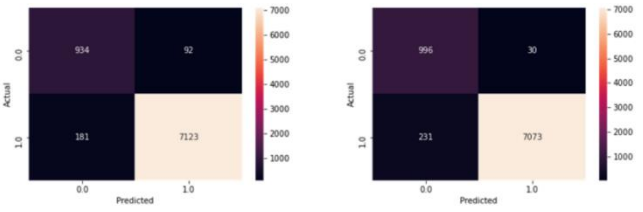

Fig. 2. Confusion matrices for RF classifier (default, hyperparameter tuning).

equally. This performed better than the distance weight function which points are defined by the inverse of their distance. Also, due to the sample size, a user-defined function was unfavourable.

Retraining the model with these hyperparameters resulted in an accuracy score of 96.6%, a precision score of 90.6%, a recall score of 95%, and an ROC-AUC value of 0.992. Comparing this with the default model, the accuracy and ROC-AUC value showed no significant change. There was a slight decrease in precision, however, the increase in recall outweighed this change and provided a slight improvement to the model overall.

Applying standardization to the features prior to training the model resulted in an accuracy score of 96.9%, a precision score of 91% and a recall score of 95.9%, all of which are improvements upon training the model with the default features. Applying normalization resulted in an accuracy score of 96.8%, a precision score of 90.8%, and a recall score of 95.9%. Whilst still an improvement, it performs slightly worse than when applying standardization.

Using the stratified k-fold cross-validation approach with K neighbors classification resulted in a mean accuracy score of 91.3%, indicating that the model will not perform as well when it comes to unseen data.

A comparison of evaluation metrics throughout the K neighbors model configuration can be seen in Table VI, Fig. 3 shows the corresponding confusion matrices.

C. Support Vector Classifier

A support vector machine (SVM) is an algorithm which aims to find a hyperplane in an N-dimensional space that distinctly classifies the data points [9]. The SVM classifier implemented was taken from the sci-kit learn class. Support vector machines can be effective with high dimensional data and works well when classes are well separated. The trade-off is that a support vector machine on a large dataset can take a while to train, as was the case with this dataset.

Training the support vector classifier using the given training and testing sets with default hyperparameters resulted in an accuracy score of 96.8%, a precision score of 90.8%, a recall score of 95.8%, and an ROC-AUC value of 0.992.

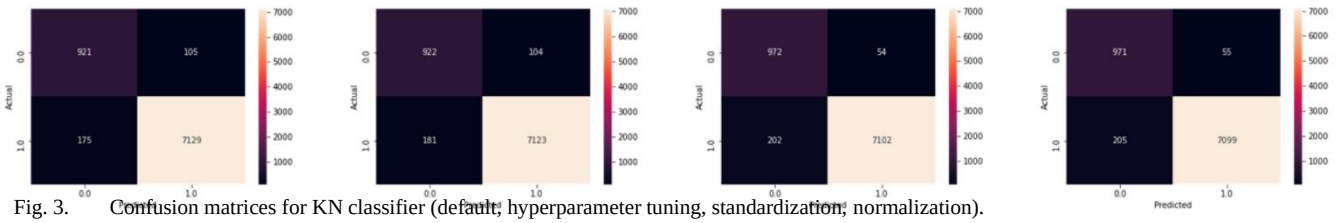

Fig. 3. Confusion matrices for KN classifier (default, hyperparameter tuning, standardization, normalization).

The hyperparameters chosen to optimise for support vector classification were C, the regularization parameter where the strength of regularization is inversely proportional to C, kernel, the kernel type to be used in the model, and gamma, the kernel coefficient. It was found that for the model to perform better at higher values of C, the value of gamma had to be reduced. The eventual value set for C was 1000, with gamma being set at 0.0001. The kernel used was 'rbf' as this seemed to outperform other kernels such as 'linear' and 'sigmoid'.

Retraining the model with these hyperparameters resulted in an accuracy score of 96.9%, a precision score of 90.6%, a recall score of 96.6%, and an ROC-AUC value of 0.992. Comparing this with the default model, the ROC-AUC value did not change. There was a slight decrease in precision score, along with a slight increase in the accuracy score. The recall score saw a more significant increase. Overall, the chosen hyperparameters led to an improvement in the models predictions.

Applying standardization to the features prior to training the model resulted in an accuracy score of 96.8%, a precision score of 90.3% and a recall score of 96.9%. With a slight decrease in accuracy and precision, and a slight increase in recall, it can be seen that standardization has not had a significant enough impact to warrant its use. Normalization produced equivalent performance in each of these metrics.

Using the stratified k-fold cross-validation approach with support vector classification resulted in a mean accuracy score of 96.6%, showing the same indication as random forest classification, that the data will still perform well on average.

A comparison of evaluation metrics throughout the support vector model configuration can be seen in Table VII, Fig. 4 shows the corresponding confusion matrices.

IV. MODEL COMPARISON

All three classification models performed well with the given dataset with the lowest accuracy recorded after tuning hyperparameters being 96.6%. The highest accuracy recorded was 96.9% which was achieve by both the K neighbors classifier and the support vector classifier.

The support vector classifier displayed a noticeable computational time difference in comparison to the other two, taking significantly longer for training. This was likely due to the hyperparameters used, in particular the high regularization parameter.

With a high recall score for each model, it can be seen that most of the positive cases in the test set will be indeed be labelled positive. Each model also displayed a lower precision rate, indicating that there is a slightly lower chance that a result predicted positive is actually positive. However,

TABLE V. EVALUATION METRICS FOR RF

	Accuracy	Precision	Recall	AUC-ROC
Default	96.7%	91.2%	94.1%	0.996
Hyperparameter Tuning	96.8%	90.3%	96.9%	0.993
Standardization	96.8%	90.3%	96.9%	NA
Normalization	96.8%	90.3%	96.9%	NA
Stratified K-Fold CV	96.6%	NA	NA	NA

TABLE VI. EVALUATION METRICS FOR KN

	Accuracy	Precision	Recall	AUC-ROC
Default	96.6%	91.2%	93.6%	0.992
Hyperparameter Tuning	96.6%	90.6%	95.0%	0.992
Standardization	96.9%	91.0%	95.9%	NA
Normalization	96.8%	90.8%	95.9%	NA
Stratified K-Fold CV	91.30%	NA	NA	NA

TABLE VII. EVALUATION METRICS FOR SVC

	Accuracy	Precision	Recall	AUC-ROC
Default	96.8%	90.8%	95.8%	0.992
Hyperparameter Tuning	96.9%	90.6%	96.6%	0.992
Standardization	96.8%	90.3%	96.9%	NA
Normalization	96.8%	90.3%	96.9%	NA
Stratified K-Fold CV	96.6%	NA	NA	NA

the precision was still above 90% for all of the models trained, so this is still an acceptable value.

AUC_ROC scores recorded were all close to 1 meaning that each individual model has an ideal measure of seperability, in other words, highly likely to be able to distinguish between the two outcomes.

Both the random forest classifier and support vector classifier performed well when trained using stratified k-fold cross-validation, indicative that they are highly likely to perform well on unseen data. The K neighbors classifier did not show this same level of accuracy, however, still achieved a reasonable score and is still likely to achieve a good performance.

None of the classifications produced a considerably high percentage of false negatives or false positives, demonstrated by the performance metrics and confusion matrices.

V. CONCLUSION

All of the performance metrics show that the three classification models used perform very well on the given dataset and our likely to correctly predict the outcome for any given case. Of course, there is still slight room for improvement with each.

The dataset used consisted of a high ratio of positive outcomes in comparison to negative. It was decided against resampling the dataset to overcome this due to the high possibility of selection bias in the future.

Considering the sensitivity of the outcome in question, future improvements to the models, which could come from a higher number of complete samples for which to train them on, would likely be needed in order for the models to be of use.

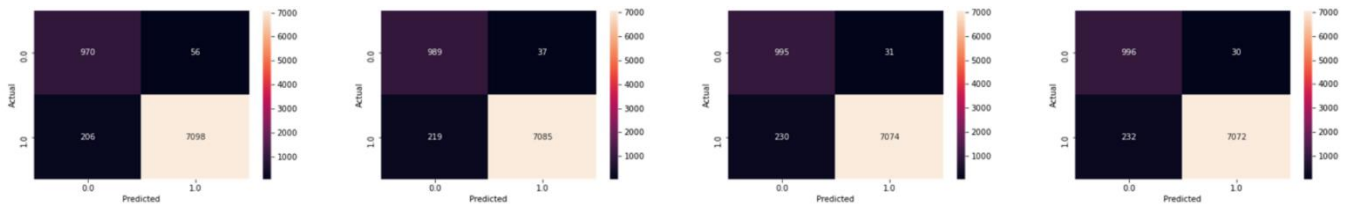

Fig. 4. Confusion matrices for SVM classifier (default, hyperparameter tuning, standardization, normalization).

REFERENCES

[1] *WHO Coronavirus (COVID-19) Dashboard* (2021). Available at: https://covid19.who.int/ (Accessed: 1 April 2021).

[2] *beoutbreakprepared/nCoV2019* (2021). Available at: https://github.com/beoutbreakprepared/nCoV2019/tree/master/latest_data (Accessed: 13 March 2021).

[3] Xu, B. et al. (2020) "Epidemiological data from the COVID-19 outbreak, real-time case information", *Scientific Data*, 7(1). doi: 10.1038/s41597-020-0448-0.

[4] *What is One Hot Encoding and How to Do It* (2021). Available at: https://medium.com/@michaeldelsole/what-is-one-hot-encoding-and-how-to-do-it-f0ae272f1179 (Accessed: 11 April 2021).

[5] *What is the Variance Inflation Factor (VIF)* (2021). Available at: https://medium.com/@analyttica/what-is-the-variance-inflation-factor-vif-d1dc12bb9cf5 (Accessed: 18 April 2021).

[6] Brownlee, J. (2021) *Iterative Imputation for Missing Values in Machine Learning*, *Machine Learning Mastery*. Available at: https://machinelearningmastery.com/iterative-imputation-for-missing-values-in-machine-learning/ (Accessed: 26 April 2021).

[7] *Chapter 5: Random Forest Classifier* (2021). Available at: https://medium.com/machine-learning-101/chapter-5-random-forest-classifier-56dc7425c3e1 (Accessed: 6 April 2021).

[8] *Day (11) — Machine Learning — Using KNN (K Nearest Neighbors) with scikit-learn* (2021). Available at: https://medium.com/@kbrook10/day-11-machine-learning-using-knn-k-nearest-neighbors-with-scikit-learn-350c3a1402e6#:~:text=KNN%20is%20a%20classification%20algorithm,and%20it%20has%20few%20parameters (Accessed: 16 April 2021).

[9] *Support Vector Machine — Introduction to Machine Learning Algorithms* (2021). Available at: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (Accessed: 12 April 2021).