Predicting Outcome of Covid-19 Patients using Machine Learning

Introduction:

In this study three machine learning models were trained and used to predict the outcome of a patient with covid-19. A dataset of over 2.6 million patients confirmed to have covid-19 by a PCR test was used, with over 100,000 labelled samples containing information on whether a patient recovered or died from the virus. The other features in the dataset, including medical and demographic information of patients could be used to predict this. Commonly used binary classification algorithms logistic regression, k nearest neighbours and random forest were trained on pre-processed data to create predictive models that could be used to estimate whether a patient had high mortality risk or not.

Artificial intelligence has already been proven to be useful in making medical predictions, helping care staff make decisions more confidently and effectively [1]. It is hoped these machine learning models could help caregivers prioritize patients that need care the most, especially during busy times where there is not adequate medical facilities for all patients. This could reduce delays in providing treatment for those that need it the most, and therefore reduce the number of deaths in hospitals that hold many covid-19 patients.

Data Preparation:

The full dataset of epidemiological information used in this experiment can be found at [2], containing over 2.6 million rows and 30 features. Before being used to train machine learning models unlabeled data needed to be dropped, categorical variables encoded and missing values dealt with.

In the first data preparation step, rows with missing values for the outcome feature were dropped as these were not labelled. I then encoded outcome into 0- deceased, or 1- recovered/discharged. Rows where the outcome variable had not made it clear whether a patient had recovered or survived were dropped. This left just over 103,000 labelled samples. I then dropped columns containing definitely useless information like admin_id and sequence_available to leave me with just potential features.

I split encoding, dealing with missing values and feature selection into two parts to make these processes easier. I first dealt with numeric columns and binary categorical columns that were easily converted to 0's and 1's (e.g. sex). I also cleaned up numeric columns with inconsistent formatting (e.g. age).

I then dealt with missing values for numeric features. I visualized the distribution, frequency and correlation of missing values using missingno to help decide on imputation techniques. Missing values were far more common in data points from some countries compared to others, however within these subsets data appeared to be missing randomly. This was especially prelevant for age and sex, indicating these values were missing at random as opposed to missing completely at random. To deal with this I implemented a random forest regression model that used the other features in the

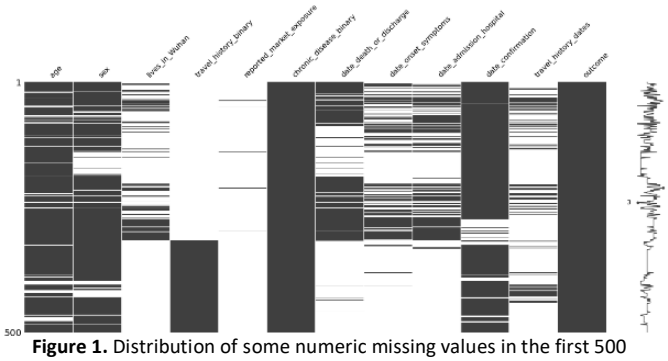

Figure 1. Distribution of some numeric missing values in the first 500 rows of the dataset, other parts of the dataset showed different amounts of missing values but similar patterns

dataset to predict age and sex. I decided not to round the continuous values for sex produced by the model as this is shown to have better performance [3]. For other columns I used simple imputation techniques like mode imputation and 0 imputation (e.g. if there was no value for travel_history_binary I assumed there was no travel history).

Next, I apply feature scaling to the age column as this can increase performance of gradient descent and distance based algorithms like logistic regression and k nearest neighbours [5]. I apply standard scaling as age roughly follows a normal distribution.

After imputation and scaling I selected numerical features to be used in the model. I used a combination of the number of missing values I had to impute, correlation with the output variable outcome and intuition to decide on which features to keep. The final list of numeric features used in the model was: age, sex, chronic_ disease_binary and travel_history_binary.

At this stage in pre-processing I carried out dataset balancing. As there were many more data points where a patient recovered (outcome=1) than died (outcome=0) I randomly down-sampled the recovered patients so there was an equal number of 1's and 0's in the outcome column of the dataset. This not only enabled me to use accuracy as the main measure to score my models, but helped me create models that weren't biased due to class imbalance [4]. I carried out balancing before encoding categorical variables as it decreased workload to manually one hot encode symptoms and chronic disease.

I then dealt with categorical variables using similar methodology to that used with numeric ones. I first dropped all location variables except province, city and country. These contained all relevant location information, but unlike other location features had very few missing values and were already standardized.

So my models could make use of categorical variables used I one hot encoding. I decided to manually one hot encode symptoms and chronic_disease due to the inconsistent nature of entries in these columns. This allowed me to research medical terms and ensure that different names for the same symptom/disease were not counted as multiple different symptoms/diseases (e.g. dyspnea and shortness of breath). I also grouped symptoms and diseases where I though appropriate (e.g. cardiac_diseases) to try and stop the number of features becoming too high. I then used pandas to one hot encode other categorical variables as formatting was more consistent in these columns.

I then dealt with missing values, for symptoms and chronic_disease I assumed a missing value meant no symptoms or disease. With missing values in country, city and province columns that appeared to be distributed randomly I used k nearest neighbour imputation. This ensured any missing values were filled in a valid way using their neighbouring data points. For example I didn't want a city value filled in that wasn't actually in the country of that data point, which is a risk with simple imputation.

Lastly, I decided which categorical features to keep based on number of missing values, intuition and model performance. The final list of categorical variables used was symptoms, chronic_disease, province, city and country. I experimented with also keeping travel_history_location, but found its impact on model accuracy was negligible, suggesting it did not encode a lot on information on outcome.

The final dataset used to train my machine learning algorithms had 11,998 entries and 864 columns. This was split into a train (≈70%), test(≈15%) and validation set (≈15%).

Algorithm Selection:

As this was a binary classification task I chose three commonly used machine learning algorithms for classification, that were also sophisticated enough to solve the problem.

Logistic regression uses the sigmoid function and log odds to make classifications from a linear equation input. It is quite simple but proven to be very effective for classification, especially the binary classification in this study [6]. k nearest neighbours (kNN) uses training examples to find the k closest neighbours in test sets, and classifies by majority vote of these neighbours. It has a very fast training time and does not require knowledge on distribution of data to be effective in classification [7]. Random forest algorithms use multiple decision trees that each separate data into similar categories. In classification the final prediction is the majority vote of these decision trees, this gives an effective prediction that is better than a single decision tree. Random forest is widely used for classification problems and shown to be effective at tackling them [8].

I also experimented with support vector machines in my study, deciding not to use them. Despite comparable accuracy to other algorithms the training time was considerably longer, making hyperparameter tuning and cross validation very difficult.

Hyperparameter tuning/model evaluation:

Before evaluating success of final models I tuned hyperparameters on the validation set to try and make each algorithm as effective as possible. I selected the most important hyperparameters [9] for each model and grid searched within reasonable ranges using scikit-learn. This allowed me to find the hyperparameter combination that produced the highest accuracy on the training set for each algorithm. I make sure not to use the validation set for training, testing or cross validation to protect against overfitting.

After hyperparameter tuning I tested each of my models on my chosen performance metric- accuracy. Initial experiments show random forest predicting with the highest accuracy on the test set, followed by logistic regression then kNN. I also look at performance of models on the training set, to try and detect if my models are overfitting. Comparable accuracy on train and test sets is an indicator models are not overfitting despite my fairly large number of features.

```
logistic regression model accuracy score on training set:  0.7118149614503021
logistic regression model accuracy score on test set:  0.70625

kNN model accuracy score on training set:  0.677536986872265
K nearest neighbours model accuracy score on test set:  0.6783333333333333

RF model accuracy score on training set:  0.7258803917482809
random forest model accuracy score on test set:  0.71125
```

Figure 2. Performance of models on test and training set

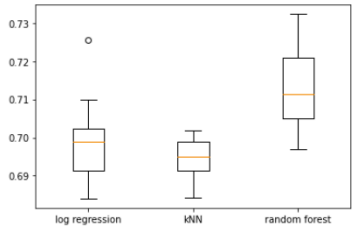

Figure 3. Box plot of cross validation accuracy scores

I then used 10 fold cross validation to get a more reliable measure of model accuracy, and as another assessment to make sure my models weren't overfitting. Results showed the same pattern as initial accuracy scores, with random forest scoring highest. However, standard deviation of scores did overlap, indicating accuracy differences could just be down to chance.

I then plot receiver operating characteristic or ROC curves for all models, which show the true and false positive rates at different classification thresholds. The area under the ROC curve or AUC is the probability a model will rank a random positive sample above a random negative. It provides a performance measure which takes into account

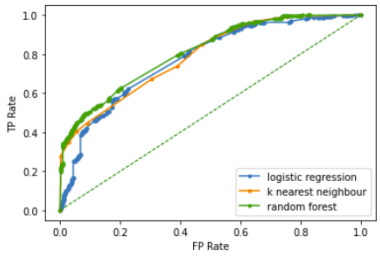

Figure 4. ROC curves for all models

```
Area under ROC curve for logistic regression:  0.7741687630529714
Area under ROC curve for k nearest neighbours:  0.7885620372128346
Area under ROC curve for random forest:  0.8104886557812796
```

false and true positive rates, and is scale-invariant, unlike accuracy [10]. In this experiment AUC, like accuracy is highest for random forest, however AUC for kNN is higher than logistic regression, suggesting that despite having lower accuracy the kNN model may be more discriminative than the logistic regression model.

Lastly, I visualize true and false negatives as well as positives with confusion matrices.

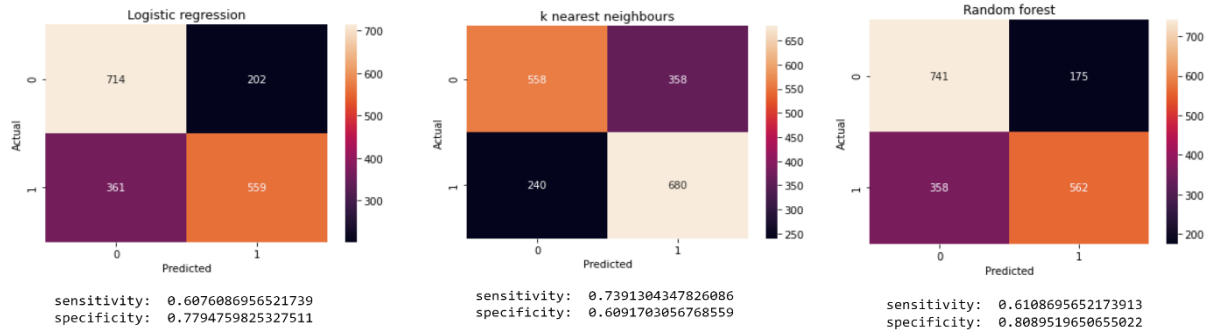

Figure 5. Confusion matrices for all models used the calculate sensitivity and specificity

Confusion matrices show k nearest neighbours has the highest sensitivity (TP/TP+FN) and random forest the highest specificity (TN/TN+FP). In this experiment specificity is probably the more important measure: it is likely giving someone extra care that doesn't need it, is less damaging than not giving care to a patient that should have had it.

Overall, the random forest model slightly outperforms kNN and logistic regression on the vast majority of measures, with an accuracy of 71% and AUC of 0.81. This is in line with papers comparing random forest performance to logistic regression and kNN [11], [12]. Furthermore, as random forest models work with subsets of data, they are very effective in tasks like this one where data has high dimensionality. kNN on the other hand may suffer from this, as it is dependent on data points being close together, which becomes rare as amount of dimensions increases. Random forest is also not held back by non-linear features, which can limit the accuracy achieved by logistic regression.

Conclusion:

A classification accuracy of 71% is not bad, but probably not high enough for use in a clinical setting. It is likely the accuracy of models was limited by large amounts of missing data. Age and sex are known to be strong indicators of outcome, yet in the dataset used to train models the vast mojority of entries for age and sex were missing. When this many values of important features are missing there is no highly effective way to deal with it, dropping values can lead to bias and large losses of information, and by imputing, even with a model based method it still leads to a great amount of error. We can never be 100% sure missing values would follow the same pattern as values present in the dataset. The similar performance of all models trained also indicates the main limiting factor of accuracy in this experiment was the data.

Nevertheless, it is entirely plausable a binary classifier to predict covid-19 outcome could be far more accurate if trained on the correct data. It is suggested to increase accuracy of binary classifiers to predict covid-19 outcome or inform decisions on other medical problems in the future, more medical institutions should be encouraged to take demographic and medical data of patients, especially in countries and cities where this is not common practice. For covid-19 specifically I recommend more information is taken on all people that test posotive, even if they do not end up hospitilised or severly ill from the virus. As with with my machine learning models and any others, the data is the most important aspect in making reliable and accurate predictions.

Reference list:

1. Chang D, Chang D, Pourhomayoun M. Risk Prediction of Critical Vital Signs for ICU Patients Using Recurrent Neural Network. In: 2019 International Conference on Computational Science and Computational Intelligence (CSCI). 2019. p. 1003–6.
2. Xu B, Gutierrez B, Mekaru S, Sewalk K, Goodwin L, Loskill A, et al. Epidemiological data from the COVID-19 outbreak, real-time case information. Scientific Data. 2020 Mar 24;7(1):106.
3. Allison P. Imputation of categorical variables with PROC MI. 2005 Jan 1;
4. Pourhomayoun M, Shakibi M. Predicting mortality risk in patients with COVID-19 using machine learning to help medical decision-making. Smart Health (Amst). 2021 Apr;20:100178.
5. Bhandari A. Feature Scaling | Standardization Vs Normalization [Internet]. Analytics Vidhya. 2020 [cited 2021 May 12]. Available from: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/
6. Wright RE. Logistic regression. In: Reading and understanding multivariate statistics. Washington, DC, US: American Psychological Association; 1995. p. 217–44.
7. Peterson LE. K-nearest neighbor. Scholarpedia. 2009 Feb 21;4(2):1883.
8. Pal M. Random forest classifier for remote sensing classification. International Journal of Remote Sensing. 2005 Jan 1;26(1):217–22.
9. Brownlee J. Tune Hyperparameters for Classification Machine Learning Algorithms [Internet]. Machine Learning Mastery. 2019 [cited 2021 May 6]. Available from: https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/
10. Classification: ROC Curve and AUC | Machine Learning Crash Course [Internet]. Google Developers. [cited 2021 May 7]. Available from: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc
11. Couronné R, Probst P, Boulesteix A-L. Random forest versus logistic regression: a large-scale benchmark experiment. BMC Bioinformatics. 2018 Jul 17;19(1):270.
12. Thanh Noi P, Kappas M. Comparison of Random Forest, k-Nearest Neighbor, and Support Vector Machine Classifiers for Land Cover Classification Using Sentinel-2 Imagery. Sensors (Basel) [Internet]. 2017 Dec 22 [cited 2021 May 12];18(1). Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5796274/