Covid-19 ML Outcome Report

skww86
Computer Science
Durham University
Durham, United Kingdom
skww86@durham.ac.uk

I. Introduction

In this report, we will be exploring the Covid-19 dataset [1] which contains epidemiological data from the Covid-19 pandemic. The dataset contains a row for every confirmed case of Covid-19 across the world. For each case, data related to the infected person / additional information related to the case is provided where available, such as the age of the person. One column states the 'outcome', where available; this usually describes whether the person was discharged from hospital, died as a result of Covid-19, or similar. The problem I wish to solve is to classify whether or not someone who was infected from Covid-19 survived or died as a result, based on information about the person such as their age, gender, location etc. which I will describe in further detail later. This could be extremely important information to public health bodies; if they are able to predict whether or not someone who was newly infected with Covid-19 is likely to eventually die as a result, they can quickly take action to send them to hospital and prioritise health care for that person in an attempt to hopefully prevent their death. Throughout this report, we will refer to whether someone survives or dies as the case's 'outcome'.

II. Data Cleaning

The original dataset contains a large number of columns, some of which contain information which is not relevant to a model that is trying to predict the outcome. We start off by deleting any such columns. There are also some columns that could be considered to have relevant data, but we delete these columns too because they either do not have enough data for enough rows, or we deemed them unnecessary in the interests of a more simplistic model. The columns we retain are age, sex, province (which acts as our geographical indicator), the confirmed date of the case, a binary travel history column (i.e. did the person recently travel), a binary chronic disease column (i.e. does the person have a chronic disease), and of course, outcome, which is the column we are trying to predict. We then remove any rows where any of these columns do not have a value available, as the data will be required for our predictive models. Then, we start cleaning the columns. The outcome column currently contains a range of different words/phrases, such as 'discharged', 'severe', 'died' etc. My intention is to convert this into a binary dead/alive column, so that we can create a binary classification problem. To achieve this, I initially scanned the dataset and made a list of all the unique words/phrases that could be found in this column. I then partitioned these words/phrases into 3 lists: those that implied the person died, those that implied the person survived, and those which were unclear and could have been either. Any rows where the value was unclear were deleted. Any rows where the value was a word/phrase associated to death had the value changed to 'dead', and likewise, any rows where the value was associated to survival had the value changed to 'alive'. Thus, the column now only contained two unique values. Other columns need to be cleaned too. For example, the Age column often contained a range of values rather than a specific value. Wherever this occurred, I changed the value to the middle of the range, so for example, if the range was 30-40, I would set the value to 35. It was also necessary to ensure all values were numeric as some were initially stored as a string.

III. Encoding and feature transformation

We recall that one of the columns we are using contains the date that the case was confirmed. In order to make these values conducive to the machine learning model, we encode them by converting the data to a number; namely, the number of days that have elapsed until that date since 01/01/2020. That way, the figure is still representative of the date, but it is now in a numerical format that is simple for any machine learning model to compare and process. We perform one hot encoding on the other columns so that they are also in a format for the machine learning model to process. For most of the columns, they only originally took 2 possible values anyway (e.g. sex taking male or female) so one column remains after the one hot encoding (in this example, 'sex_male'). But there are multiple different possible provinces, so a new column is generated for each possible province.

IV. Machine Learning Algorithms

As described before, this is now a binary classification problem, so I needed to pick 3 machine learning algorithms that are appropriate to solve such a problem. The selected algorithms also needed to be somewhat appropriate for the nature of the dataset we are using and its content. The 3 algorithms I selected were: Logistic Regression, K-nearest neighbors and (linear) SVM. Since I will be comparing various factors between the algorithm models, I wanted to select 3 algorithms that were all unique in their own ways and were not too similar, in the hope that I might get different results which yield interesting comparisons. So, I selected Logistic Regression as it is a high bias / low variance algorithm, and K-nearest neighbors as it is a low bias / high variance algorithm, so there is some variety there. Bloom et al. [2] showed, using different statistical learning methods on real world datasets, that generally Logistic Regression has a higher interpretability and a lower accuracy, whereas SVMs were the opposite and had a lower interpretability and a higher accuracy, whereas K-nearest neighbors was somewhere in the middle, so there was a good degree of variety between the algorithms in that sense.

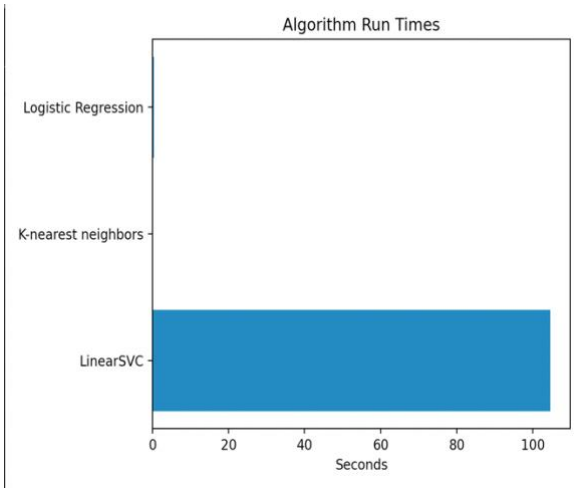
V. Setup

Our (now cleaned and transformed) dataset is unbalanced in the sense that only around 24% of the rows correspond to a 'dead' outcome, whilst all the others correspond to an 'alive' outcome. This means that if we were to randomly partition the rows into train and test sets, there could be a very large difference in the proportion of 'dead' outcomes

between the two sets. To solve this, we implement a stratified split to ensure that the ratio of 'dead' outcomes to 'alive' outcomes in both the train and test sets are more or less equal – and equal to that of their ratio in the original set. There is still randomness in the selection and partition of the sets – which we have selected to be 70% train and 30% test. The 3 algorithms Logistic Regression, K-nearest neighbors and Linear SVC are fitted to the training data and used to create 3 independent predictive models, whose predictions we shall now analyse and compare. It is worth noting that we have manually given Logistic Regression and Linear SVC a max_iter flag because without this they were reaching their iteration limit.

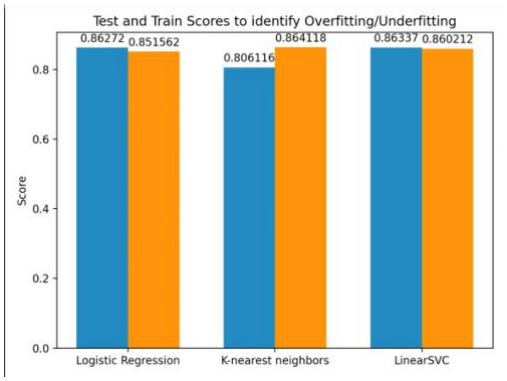
VI. Comparisons

After having let the three predictive models run and predict the outcomes in the test data set, we now analyse the results. Firstly, we analyse the run time of each algorithm – i.e. the time it took for each model to fit the train and test sets and then predict the outcomes in the test set. The results are visualised below:



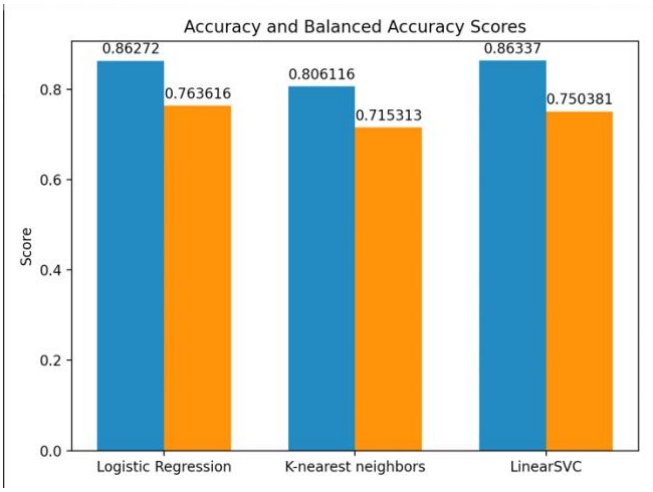
As you can see, the Logistic Regression and K-nearest neighbours ran very quickly, finishing in under a second (with K-nearest neighbours running slightly quicker). Linear SVC on the other hand took over 100 seconds to run. This is still an acceptable running time in my opinion, but perhaps the reason why it took longer than the others is because SVM classifiers don't scale so easily [3], a fact mentioned in the SVC documentation.

Next, we analysed the scores of each algorithm on both the test sets, and the train sets, in order to identify whether or not any of the algorithms are underfitting/overfitting – if there is a significant difference between the two scores this would imply either underfitting/overfitting.



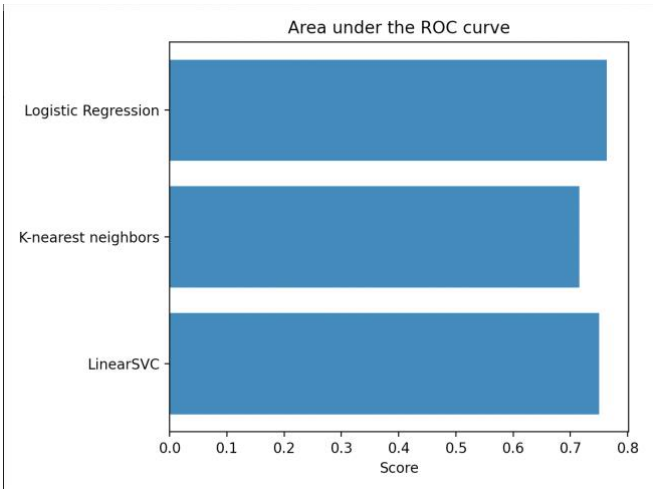
In this bar chart, the orange bars represent the algorithms' respective scores on the train set, whereas the blue bars represent the algorithms' respective scores on the test set. As you can see, the scores between the two bars for Logistic Regression and Linear SVC are almost identical (just a very minor degree of underfitting, but not to the extent that it is really worth noting). This shows that they perform well in the sense that they are not overfitting/underfitting. On the other hand, K-nearest neighbours has a score on the train set that is roughly 0.6 greater than its score on the test score. Thus, K-nearest neighbours is showing some overfitting tendencies. It isn't a major overfitting, but it is certainly worth noting. We are using a k value of 5 for our k-nearest neighbours algorithm. Perhaps, in order to reduce these overfitting tendencies, we could have fine-tuned our value of k to a particular number where overfitting is kept to a minimum. This could have been achieved by cross validating the overfitting against multiple different values of k to find the optimal value.

Next, we analysed the accuracy, and the balanced accuracy of each algorithm's predictions. Accuracy measures the proportion of correct predictions, whereas balanced accuracy measures the accuracy where each sample is weighted according to the inverse prevalence of its true class [4] – it does this by taking into account the number of true positives, true negatives, false positives and false negatives in the predictions. It is especially useful for our data set because we are dealing with an imbalanced data set in the sense that only around 24% of the outcomes are 'dead' whereas the rest are 'alive'. Thus, measuring accuracy alone might not be entirely trustworthy. In theory there is a class_weight = balanced parameter which forces the predictive models to consider this whilst making its predictions, but this parameter is only available for Logistic Regression and Linear SVC, and not K-nearest neighbour, so I opted not to use it as I wanted my parameters to be as consistent across the 3 algorithms as possible in the interests of a fair test.



The blue bars here represent the accuracy, whereas the orange bars represent the balanced accuracy. All 3 algorithms perform pretty well on this dataset, with Logistic Regression and Linear SVC performing very similarly. They are also both more accurate than K-nearest neighbours, whose accuracy and balanced accuracy is worse. Again, it might have been possible that a different k value for our K-nearest neighbours algorithm would have yielded a better performance. It is no surprise that all 3 algorithms have a worse balanced accuracy score than accuracy; this would be due to the imbalanced nature of our data set. Interestingly, despite being the worst in both metrics, the reduction in score between accuracy and balanced accuracy is lowest for K-nearest neighbours.

Next, we analysed the area under the ROC curve for each algorithm. This metric represents the probability that the classifier will rank any given positive example higher than any given negative example. Thus, it is very useful in the sense that it gives us a general idea of how it might rank predictions to be one outcome over the other. The closer to 1 the value is, the better as it means the model has a good measure of separability.



All three algorithms have fairly similar values, with Logistic Regression edging Linear SVC in this particular metric - but again, K-nearest neighbors is slightly worse than the other two. This is in line with its performance so far in the other metrics. I would expect the value of the area under the ROC curve to have a fairly significant correlation with accuracy metrics.

VII. Conclusion

We have established a binary classification problem to predict the outcomes of confirmed Covid-19 cases. This was achieved by identifying several key columns deemed to contain relevant data towards establishing such an outcome, and deleting all rows where such data was unavailable. The outcome column itself was simplified into taking only 2 distinct values by either converting or deleting rows based on the current value. One thing I have learnt from this process is that by having this level of strictness in the data requirements, we ended up deleting the majority of the rows in the original dataset. The downside of this is that our final findings and predictions are not necessarily fully representative of the entire dataset. If I were to do a similar project to this in the future, I would find a compromise between ensuring I have sufficient data and retaining as many rows as possible. This could be achieved by say, populating blank values in numerical columns with the mean of the other values in that column. We have built three different predictive models based on 3 different algorithms to solve our binary classification problem: Logistic Regression, K-nearest neighbors, and Linear SVC. We have used several metrics to analyze and compare the results of these predictive models. Generally speaking, all three predictive models yielded decent results. The optimal selection based on performance would be either Logistic Regression or Linear SVC, and if runtime was then taken into consideration, Logistic Regression would be the preferable option over Linear SVC. K-nearest neighbor did not perform as well as the other two, but this did raise an important point to me. Given the significance of the k value that is chosen to be used in the implementation (5 in our case), it is arguably very important to initially cross validate each algorithm with varying parameters to see which ones it performs optimally with. We can conclude that, with the parameters that we used for each algorithm, that Logistic Regression was the optimal choice out of the three, but we cannot assume that Logistic Regression is objectively better than the others for this particular data set – as there could be say, a different k value for k-Nearest Neighbors that causes drastic performance improvements. As I gain more experience with using different machine learning algorithms with different parameters, I would hope that in the future I could initially select a range of potentially viable parameters to initially cross validate each algorithm, so that I can be confident each algorithm has been equipped with the best parameters relative to the dataset in question, and therefore make objective conclusions when comparing them.

REFERENCES

[1] Xu et al., “Epidemiological data from the COVID-19 outbreak, real-time case information” https://www.nature.com/articles/s41597-020-0448-0

[2] J. Bloom, H. Brink, Overcoming the Barriers to Production-Ready Machine Learning Workflows

[3] https://ogrisel.github.io/scikit-learn.org/sklearn-tutorial/modules/generated/sklearn.svm.SVC.html

[4] https://scikit-learn.org/stable/modules/model_evaluation.html#balanced-accuracy-score