Machine Learning Report

chpf93
Department of Computer Science
Durham University
Durham, United Kingdom
chpf93@durham.ac.uk

***Abstract*—In this report I experiment with a variety of machine learning models using a data set consisting of known COVID-19 patients and attempt to predict the outcome of a patient's infection.**

I. Problem Framing

In this project I am going to explore a data set related to the ongoing COVID-19 pandemic which has resulted in over 125,000[1] deaths in the United Kingdom alone [1]. The data set which I will be using contains records relating to the outcome of individual coronavirus patients containing key information such as the age, location, travel history, sex and chronic disease history [2]. This is continuously updated from a multitude of sources around the world and provides detailed data about individual cases. Throughout this project, I will be using the data set downloaded from their GitHub repository[2] which was accessed on 12/04/2021.

In this project, my aim is to predict the outcome of a COVID-19 patient using the data set provided. Predicting the outcome of COVID-19 could have many applications such as considering more invasive treatment options earlier or prioritising treatment if health services are overwhelmed. To do this, I will use a binary classifier predicting whether a patient will survive the virus (the positive class) or if they will unfortunately die as a result of the infection (the negative class). I will use supervised learning techniques to train the models by using instances labelled with the outcomes.

The implementation will be in Python 3.8.2 in Jupyter Notebook format with pandas 1.1.3, numpy 1.19.2, matplotlib 3.3.2 and scikit-learn 0.23.2 for the data analysis and implementation of the machine learning models which will be used to classify the instances.

II. Data Cleaning

Before I am able to use the data set to train machine learning models I need to clean the data. I used pandas [3] to do this because it has native support for loading large CSV files and manipulating the data via the data frames that it provides. This includes handling missing data, one-hot encoding categories, grouping data and imputing values where possible. Initially, there are 33 columns and 2,676,311 records.

There is a significant amount of missing data in important columns. As my project is related to predicting outcomes I require instances with the outcome available. Without this, I will be unable to train and evaluate my model. Imputation of these values would make the results unreliable since they are being used to evaluate the model. There are 2,368,929 records without an outcome. This is a large portion of the data set but removal of these records is necessary and still leaves a substantial amount of data available to use.

It was also important to try and impute some values for columns if possible to restore some of the data. I did this for both the 'chronic_disease_binary' and the 'travel_history_binary' columns. Focusing on the 'travel_history_binary', I observed that there were 93 records which were missing the binary identifier but had either dates or location of travel which indicated that there was a history of travel. As such, I used these values to restore the 'travel_history_binary' for these records. It was a similar scenario for the chronic disease entries.

I also used pandas to one-hot encode the province and cities. To do this, I looked at all of the provinces and cities and chose to keep all of those that are used by at least 1% of the data set. All of the others - and missing values - were grouped into an out-of-vocabulary bin labelled as 'other'. I did this separately for provinces and cities and prefixed the columns appropriately. The reason that I used one-hot encoding was because this data was nominal and it would be incorrect to represent this as numerical identifiers for each value as this could be misinterpreted by machine learning models that look for patterns in the numerical data. One-hot encoding was the more appropriate technique. By limiting it to those with at least a 1% share of the data set it prevented the number of new columns introduced becoming excessive.

The outcomes recorded in the data were not uniform across the records. As such, I mapped the outcomes to a boolean (labelled 'has_survived') set to true if they survived and false if they have died.

Finally, I process the ages and sexes. Approximately 10% of the instances had age and sex recorded. It is well-known that age and sex are important factors affecting the outcome of a COVID-19 infection [4]. As such, it is important that these factors are included in the data set which effectively left little choice but to remove records that were missing these pieces of information otherwise the reliability of my models would be hindered. For the remaining records, I mapped the ages to categorical groups instead. These were of the form '0-9', '10-19' etc. with a final category of '80+'. These are ordinal

[1] as of 12/04/2021
[2] https://github.com/beoutbreakprepared/nCoV2019

categories since they represent ages, as such I mapped these to numbers in increasing order. I categorised the ages because the data provided was a mix of categorical and exact ages thus it was simpler to convert them all to the same format.

Overall, this left 33,401 instances with 44 features each. I split this into training and test sets with 70% of the records in the training set and the remainder in the test set.

III. Model Experimentation

A. Class Imbalance

The data set has been cleaned sufficiently and I will now utilise this to make predictions about the outcome of COVID-19 patients. First, I need to define a way to compare the models that I am going to train.

Many of the metrics discussed in lectures such as accuracy, precision and recall will be useful to provide a basic evaluation of the models. However, due to the imbalance of the classes these values could be misleading. This is because 96.1% of patients in the data set survived (i.e. the positive class) and 3.9% died (i.e. the negative class) which would skew the accuracy results. This became apparent when using scale-dependant models as discussed later. As such, I utilise a couple of additional statistics that will be useful. The first of these is the Matthews Correlation Coefficient (MCC) [5] which is a measure of the success of binary classification model. It takes into account all values from the confusion matrix and has its foundation in statistical analysis; it is regarded as well-performing metric for these imbalanced problems [6]. Another measure I will use is the Balanced Accuracy Score (BAS) which is also a measure for imbalanced binary classification problems and accounts for the impact of false predictions better than the regular accuracy [7]. On top of these new measures, I will use some other metrics discussed in lectures such as the area under the ROC curve and the F1 score which are useful ways to evaluate the performance of a classifier.

B. Support Vector Machine

I experimented with a variety of different models before selecting the three that I wanted to use fully. I will discuss a variety of models that I tested. The first model that I selected was the Support Vector Machine (SVM) model. As seen in lectures, the idea of a SVM is to maximise the margin between the two classes using a hyperplane. To begin, I naively pass the data into the SVM to get some benchmark results. The results were poor. This was expected because I had not scaled any of the data which is required for a SVM because the aim is to minimise the margin and if the data is not standardised then the scale of the data will affect the position of the hyperplane. In fact, the model predicted (that out of the 10,021 test samples) every single person would survive. Importantly, this also highlights why accuracy is not a suitable statistic for imbalanced classes because the accuracy was 96.1% which initially suggests the model performs well but it does not reflect the reality of the situation as it has misclassified all of the real negatives as false positives. This was reflected in the BAS which (when adjusted such that 0 is the worst and 1 is the best) was equal to 0.

To improve the model, I first standardised the data (since SVMs are not scale-invariant) which resulted in some of the instances correctly being classified as negatives which is an improvement. It still maintains a high accuracy of 96.8% as well as an improved BAS of 0.21 and a MCC of 0.43 Although this is an improvement it can be bettered by adjusting parameters of the model. Since the classes are imbalanced severely, scikit-learn recommended to set the class_weight parameter to balanced [8]. This will adjust the weightings for the instances in the model internally and will hopefully improve the results. As a result of this adjustment, there is a significant increase in the false negatives and the accuracy decreases to 87.0%, the BAS increases to 0.725 and the MCC decreases to 0.382. It is unclear if this leads to an overall improvement of the model and it requires more rigorous, systematic testing.

To perform this systematic experimentation, I used the GridSearchCV class from scikit-learn which performs 5-fold cross validations using all combinations of the parameters that have been provided to the search. It compares the models and determines which is the best and returns the parameters that configured the best model according to a predefined scoring criteria. Initially, I experimented with this using the default scorer provided by the SVM which is the mean accuracy. As previously discussed, this is will not be an optimal scorer for the model and this was apparent with using the grid search which led to poor results. Better results were attained when using the balanced accuracy score instead. The hyperparameters 'C' - a regularisation constant - and 'γ' - the kernel coefficient - were the two primary values to change as well as which 'kernel' was being used to fit the hyperplane and the 'class_weight'. The magnitude of these values is of more likely to have an impact than the exact numbers, as such I tried values ranging from 0.001 to 100. The exact values are found in the implementation; there were 120 combinations giving 600 separate models using 5-fold cross validation. Similarly, I also used the Matthews Correlation Coefficient as the scorer in a separate grid search with the same 600 models. Notably, the two sets of parameters differed in respect to the 'class_weight'. The MCC was maximised when it was set to default and the BAS was maximised when it was set to balanced.

I use these results to compare the BAS and MCC metrics so that I can find near-optimal parameters more efficiently for future models. To do so, I compute additional metrics to evaluate the models using the parameters determined by the grid search for each scorer and compare these to determine which scorer produces better results for my data set. Table I shows the different values for the metrics achieved when using the parameters determined using the grid search for each scoring criteria. Whilst the parameters determined from using MCC lead to a greater accuracy there were 266 false positives compared with only 121 true negatives detected (and 26 false negatives). This suggests the parameters determined are biased towards the positive class which could be explained by 'class_weight' parameter not being set to balanced affecting

the internal weightings of the model [8]. In comparison, the parameters determined using the BAS as the scorer lead to a significant increase in false negatives which totalled 929 with 68 false positives and 319 true negatives. This suggests that the MCC parameters lead to a more conservative model in comparison to the BAS parameters. Ultimately, which is better would be determined by the application that the model needed to be used for. Although there is no absolute consensus on which metric is better, a recent publication suggests that MCC may be better placed for binary classification tasks [9].

C. Random Forest

The next model that I experimented with was a Random Forest. This is also a model that was introduced in lectures. It fits a variety of decision trees on to different subsets of the data to prevent overfitting and uses these trees together. Each tree predicts the class for each instance and then the majority determines the overall class for the instance. There are a few parameters to adjust for the Random Forest. The primary ones are 'class_weight' (similar to the SVM), 'max_depth' which determines the maximum height of the decision trees, 'max_features' which determines how many features can contribute to each tree and 'n_estimators' which determines how many trees make up the forest.

Again, I used a grid search to experiment with different parameters for the model. Without standard scaling, and using BAS as the scorer, the maximum attained score was 0.776. Using MCC as the scorer, the highest score attained 0.448. I also experimented with the effects of standardising the data before it is used to train the random forest. From a theoretical perspective, I did not expect the scale to have much influence since the decision trees work on patterns in the features rather than trying to minimise a cost in a vector space where the scale would have a major impact. Using the BAS as the scorer, the maximum score attained was 0.777 and using MCC as the scorer gave 0.450. This suggests that there is little difference between using standardised data in this application for the random forest and it actually leads to slight improvements in the results with different parameters although it is negligible and could be explained by the random selection of the subsets. The parameters found by the grid search using the MCC as the scorer with scaled data were: 'class_weight' = balanced, 'max_depth' = 20, 'max_features' = all and 'n_estimators' = 100.

D. Naive Bayes

Another model that I experimented with - and ultimately determined was unsuitable - was the Gaussian Naive Bayes'. With unscaled data this model made some correct predictions but it was worse than the other models examined so far. It did not have any hyperparameters to configure and as such I ruled it out. However, I did experiment with scaled and unscaled data to examine the impact. Using scaled input, the model had an accuracy of 15.3%, MCC of 0.057 and BAS of 0.096. These measures all suggest that the model performed poorly. It incorrectly classified 8,464 of the 10,021 records as being negative instances. The results indicate that the model could be classifying inversely but in that case I would have expected the MCC to be nearer to -1 rather 0 which suggests that the model is almost randomly guessing to classifying. This effect was likely caused by the standardisation reducing correlation between features in the data set which negatively impact the model.

TABLE I
SVM: MCC AND BAS PARAMETER COMPARISON

Scorer	Acc.	Recall	MCC	BAS	F1	AUC
BAS	0.901	0.904	0.425	0.728	0.946	0.920
MCC	0.971	0.997	0.497	0.310	0.985	0.770

E. Logistic Regression

Further to this, I experimented with a Logistic Regression model which attempts to estimate the odds of a specific instance being part of each class. Without any scaling on the data, the results of this model are unsurprisingly poor since the scale of the features is very different. Since this model uses regression this has a major impact on the regression line. Using scaled data, the results are significantly improved and gives a BAS of 0.729. Using the MCC as the scorer, it attained a maximum of 0.467 with parameters: 'C' = 10, 'penalty' = 'l2', 'max_iter' = 200 and 'class_weight' = auto. Further experimentation however revealed that increasing the maximum iterations led to the model converging and increasing the MCC slightly, as such I increased the 'max_iter' parameter so that the model would converge.

F. Other Models

In my implementation I also experimented with a few other models but due to space limitations I have condensed these. The first of these was a Multi-Layer Perceptron Neural Network [10]. This gave similar results to the SVM model with similar BAS and MCC values and a similar confusion matrix. I decided not to analyse the results of this model as I was more familiar with the other models discussed previously. In addition to this, I experimented with an Ensemble Voting Classifier which combines all of the other models I have trained (SVM, Random Forest, etc.) into a voting system where they each classify the instance and then the majority determines the resulting class for the instance. This was also a quality model but it is entirely influenced by the underlying models so their impact could be explored in more depth individually rather than through an ensemble classifier.

IV. ANALYSIS OF RESULTS

In this section I will analyse the results of the SVM, Random Forest (RF) and Logistic Regression (LR) classifier models using the parameters determined previously via the grid searches with MCC as the scoring criteria. I analyse the models individually and comparatively using their confusion matrices and summary statistics found in Table V.

TABLE II
SVM CONFUSION MATRIX

Predicted \ Actual	Positive	Negative
Positive	9,608	266
Negative	26	121

Table II shows the confusion matrix for the SVM model. On initial inspection it is clear that the model correctly classifies the majority of the instances as described by the accuracy of 97.1% which is better than simply assigning all instances to the positive class. However, the model does appear to be biased in favour of the positive class in comparison to the other models. This is evidenced by the high false positive rate with only 31.2% of the true negative instances being classified as such. Furthermore, the model has few false negatives (only 26) which emphasises the bias towards the positive class in the model. This could undermine confidence in the positive results due to the conservative nature of the model. In contrast, the negative results could be trusted since 82.3% of those classified as negatives are true negatives.

Table III shows the confusion matrix for the RF model. This model classifies less instances as positive than the SVM model but as a result of the trade-off it classifies more instances as true negatives. This models classifies the least true positives correctly. The relative increase in true negatives is 61.1% but the relative increase in false negatives is 1242.3% which is a massive increase. However, this model has less false positives than the SVM model although the decrease is not as significant as the increase in false negatives. This could be explained by the decrease in true positives as well which suggests that this model is less biased towards the positive class. Unlike the SVM model it would be difficult to trust the negative cases classified by this model if it was used with unlabelled real data after training since only 35.8% of the instances classified as negative have been correctly classified.

Table IV shows the confusion matrix for the LR model. This model is quite similar to the SVM model in the results that it generates as evidenced by the similarity between their confusion matrices. However, the LR model performs slightly worse overall as it has a significant increase in the number of false negatives. There is an increase 84.6% false negatives compared to the SVM model. Of the instances classified as negative by the LR model, 72.1% are true negatives which is a decrease compared to the SVM. Nonetheless, this is still a major improvement over the RF model although it is not the best of the three. In terms of the area under the receiver operating characteristic curve (AUC), the LR model has a much higher than the the SVM model (0.930 compared with 0.770) which suggests that the LR model could be adapted with different thresholds to achieve better results in different use cases unlike the SVM model. This could allow the LR model to be used in a more versatile manner.

TABLE III
RANDOM FOREST CONFUSION MATRIX

Predicted \ Actual	Positive	Negative
Positive	9,285	192
Negative	349	195

TABLE IV
LOGISTIC REGRESSION CONFUSION MATRIX

Predicted \ Actual	Positive	Negative
Positive	9,586	263
Negative	48	124

TABLE V
MODEL STATISTICS

Model	**Acc.**	**Recall**	**MCC**	**BAS**	**F1**	**AUC**
SVM	0.971	0.997	0.497	0.310	0.985	0.770
RF	0.945	0.961	0.430	0.529	0.971	0.900
LR	0.969	0.995	0.468	0.315	0.984	0.930

V. CONCLUSION

From the analysis of the results, it is clear that the RF model performs the worst of the three. The SVM and LR model are not perfect but they are capable of making accurate predictions which are reliable and could be used to assist with decision making unlike the RF model.

During this project I have experimented with a variety of different machine learning models and I have explored the relationship between the theoretical expectations of the models and the results observed when used with a real data set. I have learnt that the preconditions and limitations of using certain models has a significant impact on the results that can be obtained as well as methods to mitigate these limitations such as scaling data to improve the results of models that rely on optimising a cost function within a vector space such as the support vector machines and logistic regression. In addition to this, I have briefly explored other models such as neural network and ensemble classifiers.

Furthermore, I have learnt a significant amount about data cleaning and how it is a major part of the machine learning process. It is important that the data is structured uniformly and data is prepared in a suitable way as the preparation could impair the machine learning models.

Throughout this, I have also learnt about the problems that imbalanced classes pose in prediction problems since one class may make up significantly less of the data set than the other which causes accidental bias in the models favouring the majority class. It makes it difficult for the model to differentiate between the two classes without a sufficiently large data set.

Overall, I have successfully used different machine learning models to make predictions about the outcomes of COVID-19 patients which was my initial aim.

REFERENCES

[1] Government of the United Kingdom. Deaths — Coronavirus in the UK. [Online]. Available: https://coronavirus.data.gov.uk/details/deaths

[2] B. Xu, B. Gutierrez, S. Mekaru, K. Sewalk, L. Goodwin, A. Loskill, E. L. Cohn, Y. Hswen, S. C. Hill, M. M. Cobo, A. E. Zarebski, S. Li, C.-H. Wu, E. Hulland, J. D. Morgan, L. Wang, K. O'Brien, S. Scarpino, J. S. Brownstein, O. G. Pybus, D. M. Pigott, and M. U. G. Kraemer, "Epidemiological data from the covid-19 outbreak, real-time case information," *Scientific Data*, vol. 7, no. 1, p. 106, Mar 2020. [Online]. Available: https://doi.org/10.1038/s41597-020-0448-0

[3] T. pandas development team. (2020, Feb.) pandas-dev/pandas: Pandas. [Online]. Available: https://doi.org/10.5281/zenodo.3509134

[4] R. Pastor-Barriuso, B. Pérez-Gómez, M. A. Hernán, M. Pérez-Olmeda, R. Yotti, J. Oteo, J. L. Sanmartín, I. León-Gómez, A. Fernández-García, P. Fernández-Navarro, I. Cruz, M. Martín, C. Delgado-Sanz, N. F. de Larrea, J. L. Paniagua, J. F. Muñoz-Montalvo, F. Blanco, A. Larrauri, M. Pollán, and , "Sars-cov-2 infection fatality risk in a nationwide seroepidemiological study," *medRxiv*, 2020. [Online]. Available: https://www.medrxiv.org/content/early/2020/08/07/2020.08.06.20169722

[5] B. Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme," *Biochimica et Biophysica Acta (BBA) - Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0005279575901099

[6] D. M. W. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2020.

[7] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The balanced accuracy and its posterior distribution," in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 3121–3124.

[8] "1.4. support vector machines." [Online]. Available: https://scikit-learn.org/stable/modules/svm.html#unbalanced-problems

[9] D. Chicco, N. Tötsch, and G. Jurman, "The matthews correlation coefficient (mcc) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation," *BioData Mining*, vol. 14, no. 1, p. 13, Feb 2021. [Online]. Available: https://doi.org/10.1186/s13040-021-00244-z

[10] F. Murtagh, "Multilayer perceptrons for classification and regression," *Neurocomputing*, vol. 2, no. 5, pp. 183–197, 1991. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0925231291900235