Machine Learning Scientific Report

1. Problem framing (10%)

Coronavirus (COVID-19) is spreading fast. Since first reported in December 2019, by mid-January 2021, it has affected more than 95 million people and killed more than 2 million people worldwide. To aid the analysis and inform public health decision making, machine learning models trained on real data can be very useful.

In this paper, we build and compare predictive models using machine learning (ML) algorithms and epidemiological data from the COVID-19 outbreak. We explore the dataset and aim to solve the problem of predicting patient outcome as either "died" or "discharged" from hospital. The motivation is that the proposed prediction model may be helpful for the quick triage of patients without having to wait for the results of additional tests such as laboratory or radiologic studies, during a pandemic when limited medical resources must be wisely allocated without hesitation.

2. Experimental procedure (35%)

We begin by exploring the dataset we use, which is by Xu et al. (2020) and is publicly available on GitHub.[1] To better understand the data, we produce a number of visualisations: we create a bar plot of up to the 10 most frequent values of each categorical attribute (e.g., *symptoms*); we create a scatter plot of the *latitude* and *longitude* of each case; and, we plot the daily and monthly number of cases globally and by country (e.g., China). From these visualisations, we get a strong overview of the spread of COVID-19 by location and by time. Next, we prepare the dataset for use by ML algorithms.

2.1 Data preparation

First, we removed irrelevant instances. These were instances that were not useful for the task. Since we were training models to predict patient outcome, irrelevant instances were those whose value of the target, *outcome*, was not one of either *died* or *discharged*. To remove such instances, we had to correct structural errors, such as inconsistent capitalisation and wording among target values. We replaced the *Died*, *dead*, or similar values with *died*; and, we replaced the *Discharged*, *recovered*, or similar values with *discharged*. Instances with target values whose meaning was dissimilar to either of these two values, including *null* values, were dropped.

Second, we kept attributes which we thought would be useful. The attributes kept, which became features, were *age*, *sex*, *latitude*, *longitude*, *date_onset_symptoms*, *symptoms*, *chronic_disease_binary*, and *travel_history_binary*. After this, we replaced the *symptoms* and *date_onset_symptoms* features with a new *symptoms_binary* feature, and cleaned up the dataset by dropping any remaining instances with missing values.

1. github.com/beoutbreakprepared/nCoV2019/

Third, we transformed numerical data. We performed data binning on the *age* feature, and created categorical values for the age ranges *0-14*, *15-34*, *35-59*, *60-79*, and *80+*. These were chosen because the *age* feature was in fact already categorical, and the latter 4 categories were the 4 most frequent. We also performed feature scaling on the *latitude* and *longitude* features by standardisation, also known as Z-score normalisation, which rescales the feature values to have zero-mean and unit-variance.

Fourth, we transformed categorical data. We performed one-hot encoding on the *age* and *sex* features since they had nominal data. Also, we encode the target values so that *discharged* corresponds to 1 and *died* corresponds to 0. The reason for performing these transformations is that ML algorithms require features to be numbers.

Finally, we randomise the resulting dataset, which has 5,206 instances, as we do not want the order of the instances, which is irrelevant, to affect the model training process; and, we split the data into a training set, validation set, and test set to prevent overfitting. We create a test set by selecting a 20% subset of the dataset so there are at least 1,000 test instances, which is large enough to yield statistically meaningful results. Since we suspect the *chronic_disease_binary* attribute to be an important feature to predict *outcome*, we chose to ensure that the the test set was representative of the overall *chronic_disease_binary* distribution by doing stratified sampling. The overall distribution was that 98.2% of values were *False* while the remaining 1.8% were *True*. We also observe that we maintain an exact balance of train and test target values, which makes it easier to train models.

2.2 Model selection

The three ML algorithms we chose to build predictive models were the logistic regression classifier, gradient boosting classifier, and support vector classifier. We chose the first two algorithms because they were demonstrated in *Exercise for Logistic Regression*[2] and the third because it was recommended by the

Using the ML Cheat Sheet (for scikit-learn)[3], we are told to try the following 3 models for our classification task: Linear SVC, KNeighbours Classifier, and SVC, where SVC stands for support vector classification.

2.3 Model training

For each of the 3 models, we iteratively train the model on the training set, evaluate the model on the validation set, and tweak the model according to the results on the validation set. In particular, we employ 4-fold cross-validation so that the size of the validation set is equal to the size of the test set. This reduces the chance of overfitting. Then, we pick the model with the set of hyperparameters that does best on the validation set and confirm the results on the test set. This approach allows us to compare different trained models in an unbiased way, by comparing model performance using the test set, which is kept apart from the training process and is therefore unseen data.

2.
3. scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

2.4 Model testing

2.5 Hyperparameter tuning

2.6 Inference/Prediction

3. Results (25%)

- Make comparisons between the 3 predictive models
- Provide necessary tables and charts to summarise and support the comparisons.

4. Discussions (20%)

4.1 Chosen models

4.2 Experimental procedure

- standardisation is commonly used by KNN, SVM, PCA, etc. but not necessary for logistic reg, tree-based, and random forest
- standardisation helps better deal with outliers, but min-max can generate smaller std; could have tried fitting model to raw, normalised, and standardised data da the ncompare their performances for the best results
- could have imputed

4.3 Limitations

- Was going to include 'country', then saw that data over represents Philippines
- Further work can include date_confirmation
- We notice we have an imbalanced data set, so we employ down sampling and up-weighting during model training.
- Larger k means less (pessimistic) bias. Leave-One-Out Cross-Validation is too computationally expesive.

5. Conclusions and lessons learnt (10%)

- Discuss the results and draw conclusions from your experimentation

References

Bo Xu, Bernardo Gutierrez, Sumiko Mekaru, Kara Sewalk, Lauren Goodwin, Alyssa Loskill, Emily Cohn, Yulin Hswen, Sarah C. Hill, Maria M Cobo, Alexander Zarebski, Sabrina Li, Chieh-Hsi Wu, Erin Hulland, Julia Morgan, Lin Wang, Katelynn O'Brien, Samuel V. Scarpino, John S. Brownstein, Oliver G. Pybus, David M. Pigott, and Moritz U. G. Kraemer. Epidemiological data from the COVID-19 outbreak, real-time case information. *Scientific Data*, 7(106), 2020. doi: doi.org/10.1038/s41597-020-0448-0.