Coronavirus Machine Learning Project

Elias Percy

***Abstract*—The following report outlines my implementation of three machine learning algorithms that aim to solve a binary classification problem in relation to COVID-19, wherein close attention has been paid to the machine learning workflow. The report includes visual and textual comparisons of the performances achieved by each model, with emphasis placed on the process of hyperparameter tuning and the handling of imbalanced data.**

***Index Terms*—Machine Learning, COVID-19, Binary classification, Epidemiological data, Rare events data**

I. Background and Motivation

A situation steeped in uncertainty, the coronavirus pandemic has disrupted society to an extent not seen in decades. This applies particularly within hospitals and healthcare more generally. An effective predictive model, one able to indicate the level of treatment that is likely required for someone who contracts the virus early on, would be of invaluable assistance to healthcare services globally. Such a predictive model could be used as a tool within hospitals to predict the potential severity a patient may face, or by individuals at home to inform them if they should seek further medical attention. It may also unveil trends and patterns interlaced amongst cases that may assist with research. This model would be economically beneficial to governments and healthcare services, helping to ensure order persists in these times of apprehension

The **beoutbreakprepared** nCoV2019 dataset [1] provides relevant information to underscore the model detailed above, with each row representing a person diagnosed with the virus. Amongst the columns is an *"outcome"* feature, describing the severity of the respective case. This column was originally to contain only the values 'recovered' or 'deceased' but due to the ad-hoc data collection process this feature has become more verbose, including phrases with varying degrees of ambiguity. In any case, this "outcome" feature is suited to act as the categorical output of a *classification* model.

Definition 1. A **classification** problem is a sub-problem of *supervised learning*, where a dataset contains some *categorical outputs* to be predicted by means of various other features in the dataset.

Specifically, the type of classification problem I aim to solve will be a **binary classification** problem, where the outcome features will only have two possible values. With regards to the dataset, the values in the "outcome" column that indicate that the patient *recovered*, or is in a good, stable condition, will be evaluated to 'safe' (represented numerically by 0). Conversely, the values synonymous with the patient dying, or being in an unstable or critical condition, are mapped to 'severe' (represented numerically by 1).

The three machine learning models that I have decided to implement for this task are Logistic Regression, Random Forest Classification, and Support Vector Classification. The majority of the machine learning methods applied in my program utilise the Scikit-learn library [2], however my code is also dependent on various other external Python libraries.

II. Data Analysis and Preprocessing

A. Establishing the Data

The dataset, in its initial state, is considerably large. First, the available features were examined using a heatmap, shown in figure 1, to visualise the number of missing values. One can immediately discern from this visualisation that for many features, the vast majority of values are missing. Of particular note is that the "outcome" feature, which is of great importance, comprised 88.5% missing values - this still, however, leaves $307,775$ available values. Since this feature was to be used for the output, it would be bad practice to impute (and imputing such a vast quantity of values would be impractical).

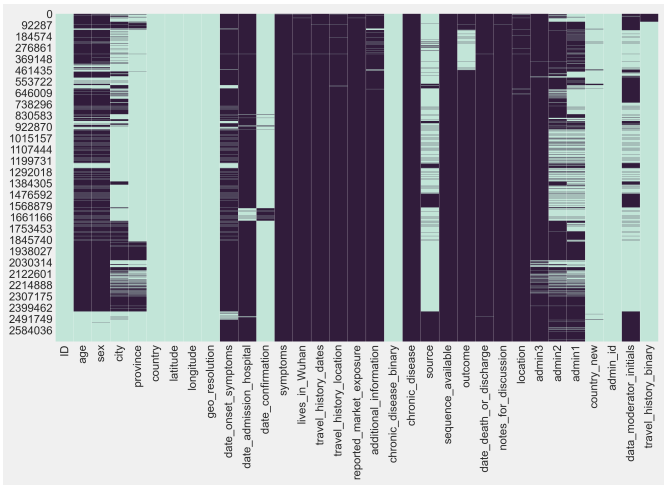

Figure 1. Heatmap showcasing missing values, with light blue indicating available values and dark blue indicating missing values.

Therefore, I developed an alternative method to read the dataset in by *chunks* of size $500,000$ rows, with any row containing a missing value for its "outcome" feature being discarded. This approach was also helpful as regards the dataset size since the burden on memory at any given time was significantly lower than when naïvely reading in the entire dataset at once.

Further analysis revealed that many columns were still mostly composed of missing values. These included features that I intuitively anticipated to be of substantial predictive value, such as "age" and "symptoms". Unfortunately, only 0.2% of the "symptoms" feature was available, rendering it useless. In any case, because I have reason to suspect that "age" will be a highly valuable predictor, it would be bad practice to impute this feature so I reduced the dataset further such that any rows missing an "age" value was discarded.

After this, the dataset now contained $34,412$ examples in total. This value is substantially lower than the original $2,676,311$, however the proportion of missing values has been drastically reduced, leading to a more valuable array of features and examples overall.

The final data establishment proceedings were to translate the values within the "outcome" column such that they are numerically represented by 0 or 1, in order to institute the data for binary classification (see def. 1), then to remove all features that were doubtlessly unhelpful either due to containing too many missing values (such as the "date_death_or_discharge" feature, comprising 99.7% missing values) or due to irrelevant content (such as the "source" feature, which wouldn't be helpful in the predictive models I intend to build as it only contained links to websites). Certain features that are likely to be removed later were kept due to their potential value during data exploration. Now, the dataset is a suitable representation of the problem domain.

B. Splitting the train and test sets

Subsequent to the preliminary measures of data establishment outlined above, it is pivotal to separate the data into distinct *training* and *testing* sets. The training set will be provided to the models for them to learn, while the testing set will be used for model performance evaluation. These partitions of the original dataset must be kept entirely separate for the duration of data exploration and preprocessing in order to evade inadvertent *overfitting*, with all preprocessing methods applied to each set at different times.

Definition 2. Overfitting is "the production of an analysis that corresponds too closely or exactly to a particular set of data." [3]

The central issue incurred by overfitting is an inability for the model to suitably fit unseen data to make accurate predictions. Splitting the testing and training sets early on is just one measure in mitigating overfitting - other methods incorporated into my program, such as cross-validation, will be discussed later on.

Intuitively, I suspected that the intrinsic predictive value of the "age" feature would be strong. Therefore, to ensure that the testing set is representative of the various age categories within the entire dataset, a *stratified sampling* approach to splitting the dataset was taken. This entailed the construction of an "age_categories" feature, whereby the numerical "age" feature (after being converted to integers - e.g., any occurrences of age ranges like '40-50' would be converted to the midpoint between them) was partitioned into 6 age groups. Then, stratified sampling was used to split the dataset while ensuring an even proportion of the different age categories appear within each set. Figure 2 illustrates the success of this method by showcasing the distribution of age categories in each set.

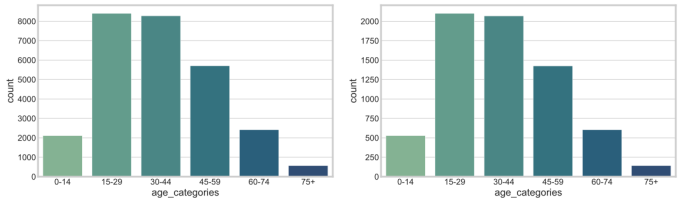

Figure 2. Distribution of age categories in the training (left) and testing (right) sets.

After splitting the dataset, the "age_categories" feature was subsequently dropped - however, its reuse was left as a potential hyperparameter. Note that I decided to use 20% of the dataset for the testing set, as this yielded a sufficiently large dataset for use in reliable evaluation of the model's performances.

C. Dataset exploratioin

After glancing at the data during the previous stages, the training set is now to be closely examined and visualised in order to garner valuable insight and discover any underlying patterns. To begin, I investigated my prediction that the "age" feature would be a valuable estimator via the violin plots showcased in figure 3. These indicate that the positive outcomes (i.e., 'severe') are more densely distributed amongst older patients, with the majority of these outcomes occurring with patients aged 60 or more. The inverse of this generally holds for the negative (i.e., 'safe') outcomes, although there is still considerable overlap and the bar chart in the same figure signifies that the age feature alone is insufficient in the prediction of outcome.

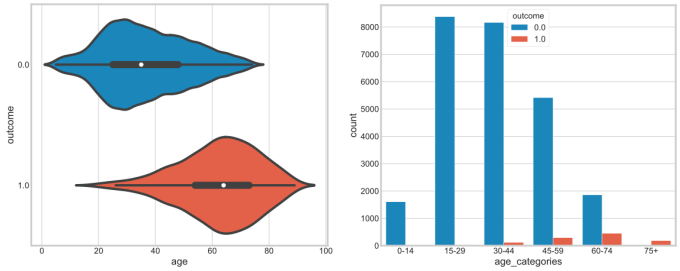

Figure 3. Violin plots (left) beside barplots (right) showcasing the relationship between age and outcome.

The bar chart is indicative of the highly *imbalanced* nature of the data, as for each age category the number of occurrences of the negative outcome consistently exceeds that of the positive outcome. In fact, the ratio of negative to positive outcomes in the data is roughly 24:1, meaning only around 4% of outcomes are of the minority class (i.e., class 1). This implies that measures are required to mitigate potential issues associated with imbalanced data.

Further discernment was acquired by geographically visualising the data, as shown in figure 4. Substantiated by this diagram is that the data is primarily distributed in India and the Philippines, with China being the third most frequently occurring country. There is a sparse distribution of data around other regions such as Africa, whilst Europe and the Americas are substantially underrepresented. To be specific, 20,432 training examples originate from India, which encompasses 84.8% of the data. This is followed by the Philippines, which appears in 3,145 training examples. Consequently, measures should be taken to mitigate the chance of overfitting (see def. 2) with respect to the underrepresented locations.

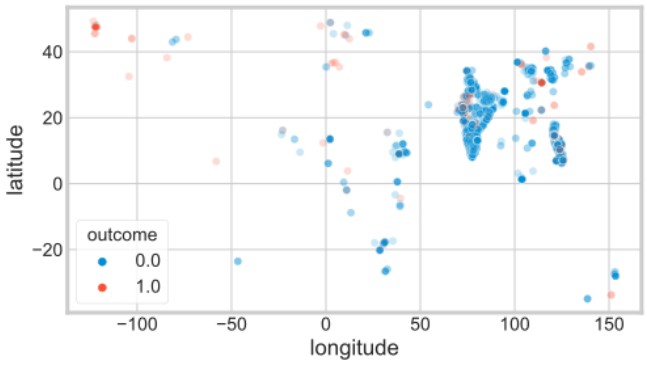

Figure 4. Scatter plot of the longitude and latitude features, with the outcome as hue.

To inspect the effect of the "chronic_disease_binary" feature on the outcome, figure 5 contains a strip plot showing firstly the distribution of outcomes with age (which has been shown to have a correlation with "outcome" in figure 3), augmented with the "chronic_disease_binary" feature. Illustrated is the pattern that 'severe' cases occur more frequently amongst patients *with* a chronic disease compared to 'safe' cases, although it is still a considerable minority of 'severe' cases wherein a chronic disease is present. Of additional note is the concentration of positive outcomes around the age of 0, which are most likely outliers that may harm the predictive ability of the models. Thus, outlier removal will appear during data preprocessing.

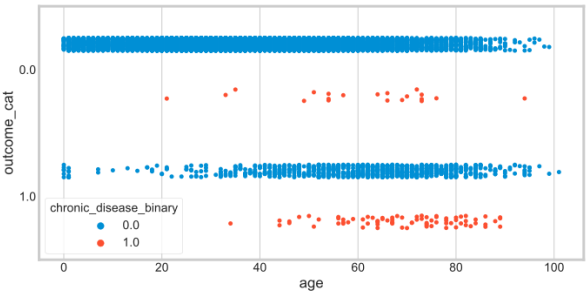

Figure 5. A strip plot to illustrate the proportion of both safe and severe outcomes where the patient had a diagnosed chronic disease.

D. Data cleaning & feature engineering

The dataset, comprising a blend of numerical and categorical features, was encoded to facilitate machine interpretation. For the binary categorical features, including "sex", "chronic_disease_binary", and "travel_history_binary", *one-hot encoding* was used, whereby for each feature the dataset was augmented to hold two additional columns, one for each binary value, with a 1 on examples where the corresponding binary value did hold, and 0 otherwise.

The "data_confirmation" feature was represented as strings, thus not having an instant numerical counterpart. The *datetime* library was used to map each date to a corresponding *Date* object, subsequently engineering a new feature entitled "days_since_start", which represented the number of days between the first recorded instance of COVID-19 and the date of that training example. The distribution thereof for each outcome is shown in figure 6. Subsequently, additional features were engineered - namely, "count_per_day", representing the number of occurrences of each date and "severe_rate", representing the proportion of *severe* cases on each date. These would assist the models in detecting any existing pattern between the

proportion of (severe) cases on a given day and the likely severity of cases more generally - intuitively, severe cases may be more likely at times of the year with certain climates. The "days_since_start" feature was clustered using k-means, placing the dates into logical groups.

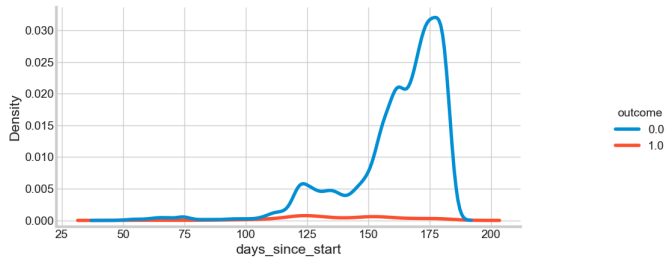

Figure 6. The distribution of dates for each outcome.

To handle the geographic data, geographic outliers (i.e., substantially underrepresented countries) were removed in order to evade overfitting in underrepresented areas. Then, the **DBSCAN** algorithm, first described in [4], was utilised to cluster the "longitude" and "latitude" features. DBSAN is a state-of-the-art unsupervised learning algorithm that performs *density-based* clustering, whereby neighbouring points are grouped together. The method handles geographic (i.e., spatial) data particularly well compared to alternative clustering techniques such as k-means.

Further measures include 'most frequent' imputation of categorical features and 'mean' imputation of numerical ones. Scaling was initially incorporated in the form of standard scaling, but other methods were trialled during hyperparameter tuning later. Most methods have been contained within a Column Transformer to allow for enhanced seamlessness and compatibility with k-Fold Cross-Validation.

III. Machine Learning

A. Introduction to the models

1) Logistic Regression (LR): Logistic Regression is a powerful model frequently used for binary classification tasks. It operates by first undertaking *regression* on a logistic function. To elaborate further, for binary classification problems, Logistic Regression models the *probability* that any given training example belongs to the output class 1 - that is, $P(X)$. If $P(X)$ is above a given threshold (usually this will be 0.5), then class 1 will be predicted as the outcome. Otherwise, class 0 is predicted. In the case of this specific dataset, class 0 refers to "safe" cases, whilst class 1 refers to "severe" cases.

To mathematically model these probabilities such that they remain between 0 and 1, the model utilise the logarithm of the odds ratio. Then, with $P(X)$ resolved to be the hypothesis function, it utilises the *sigmoid function* (also known as the logistic function) to ensure all outputs sit between 0 and 1.

2) Random Forest Classification (RFC): To understand Random Forests, one must first acquaint themselves with Decision Trees, of which Random Forests are composed. A Decision Trees comprises a sequence of 'questions' on *branches* that examine a test instance, leading to *leaves* that specify the prediction. As an ensemble learning method, Random Forests essentially train groups of Decision Trees, each on random training subsets. Then, the consensus of these Decision Trees is used by the classifier to make its decision. Often achieving high predictive capabilities, Random Forests are less susceptible to overfitting (see def. 2) than singular Decision Trees.

3) Support Vector Classification (SVC): For binary classification, Support Vector Classifiers use *kernels* to express complex decision boundaries. They separate the two classes by finding the *optimal hyperplane* (i.e., a subspace of a multidimensional space) and maximising the *margin* between the two decision boundaries - each comprising the support vectors of the two classes. The features from the input data are mapped onto the aforementioned higher-dimensional hyperplane to facilitate segregation of the initially non-linearly separable data.

B. Metrics & initial results

A multitude of evaluation metrics exists for assessing the performance of a classifier. The traditional 'accuracy' metric is very misleading with imbalanced data so the 'precision' and 'recall' scores were given more attention. The former provides an intuitive measure of a model's ability in evading the labelling of a negative instance as positive, while the latter denotes the ability of correctly labelling positive instances. From these we define the 'F_1-score' as their harmonic mean, and the modified 'F_2-score' places greater weight on the 'recall'. For an overview of these metrics applied to each model, see table I. In general, the initial performances were consistently poor. Despite occasional good precision scores, each model failed at effectively predicting the vast majority of the positive instances. Moreover, overfitting was exhibited by the SVC and RFC, which both had training recalls far greater than those attained with the testing sets (the more serious being RFC with a 45% difference).

C. Tackling imbalanced data & hyper-parameter tuning

1) Oversampling: To combat some of the issues that the imbalanced dataset facilitates, oversampling was incorporated.

Definition 3. Oversampling an imbalanced dataset entails the creation of *additional* instances of the minority class in order to compensate for the imbalance.

Specifically, I incorporated **SMOTE**, which stands for *Synthetic Minority Oversampling Technique* [5]. The method oversamples the dataset through the generation of synthetic observations following three steps: first, isolating an instance x in the minority class. Then, finding the *k-nearest neighbours* to this instance based on the feature values. Finally, selecting n of these k-nearest neighbours at random and creating n synthetic observations that would lie on a Euclidean line joining x with each of these neighbours. I utilised the 'Borderline' variant of SMOTE from the *imbalanced-learn* Python library [7], as it mitigates an issue that the generic variant has with outliers in the minority class [6].

2) Cross Validation: To tune the hyperparameters (i.e., the parameters of the models that are *not* learned) while concurrently accounting for overfitting, 5-Fold Cross Validation was used. The algorithm splits the training data into 5 'folds' of equal size at random, with 4 folds used to train the model and the additional fold used for validation. Training proceeds in this manner several more times, with a different fold for validation used during each. For a comprehensive overview of all hyperparameters tuned, consult the codebase - brief summaries will be supplied shortly.

The procedure necessitates a specified scoring metric to dictate its optimisation. Extending from the earlier exposition on metrics, 'F_2-score' was selected due to its predominant focus on the reduction in false negatives with respect to the positive class. The intuition underscoring this decision is that incorrectly predicting a patient as 'safe' when in reality they experience a 'severe' case would be more dangerous than the inverse, so maximising the recall without an influx in false negatives is desirable.

3) Additional measures: A further attempt to enhance the models included PCA dimensionality reduction, which leads to a reduction in unhelpful heterogeneity in the data. Different scaling methods were trialled, including MinMax scaling, and the removal/inclusion of certain features was assessed. In terms of the hyperparameters, these include, in addition to unique ones for each model, the number of PCA components and the sampling strategy for SMOTE.

D. Evaluation & comparisons

Regarding the optimal parameters, LR utilised the 'l2' penalty, entailing *Ridge Regression* regularization. Both LR and SVC are sensitive to their 'C-value', dictating the strength of regularization. A C-value of 206.91 was found to facilitate the optimal F_2-score for LR, while a smaller value of 0.5 worked best with SVC, meaning the SVC model necessitated stronger regularization. The SVC used the *Radial Basis Function* kernel to increase the dimensionality of the

non-linearly separable data. The RFC used the most model-specific parameters. After tuning, it utilised a forest size of 800 trees, each with a maximum depth of 80.

Figure 7 and table I illustrate comparisons of the models in terms of changes following hyperparameter tuning and the different models overall, in addition to confusion matrices in figure 8. For a more intuitive realisation of the performances, figure 7 utilises ROC curves and, for the reasons outlined in [8], Precision-Recall curves.

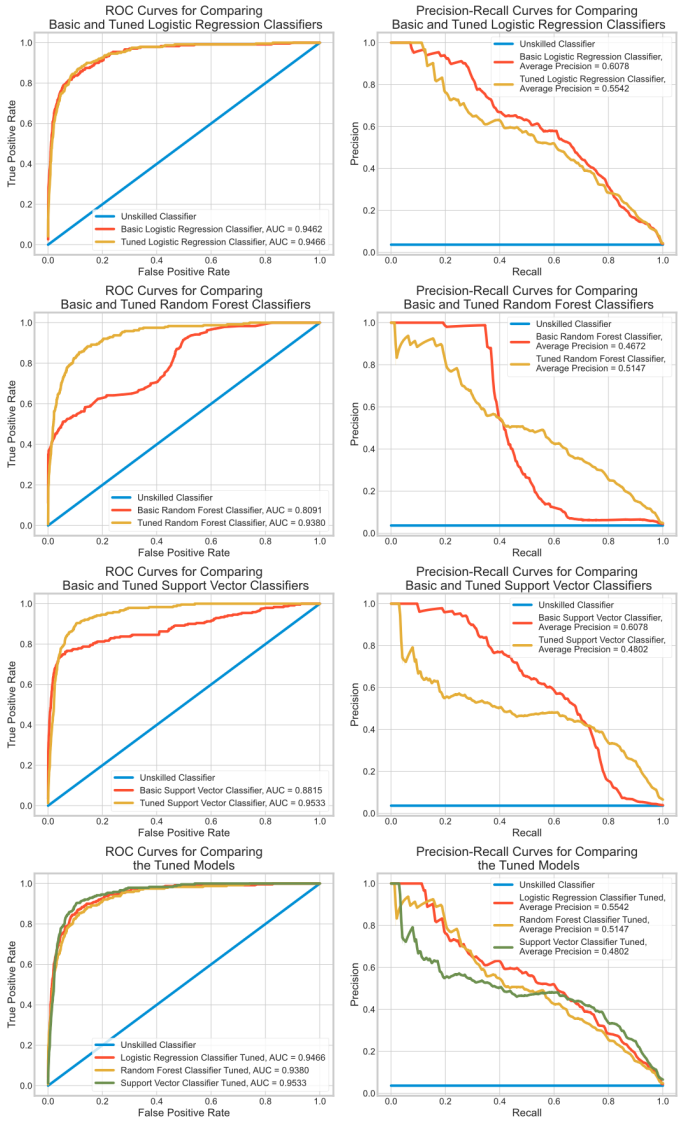

Figure 7. Receiver Operating Characteristics and Precision-Recall curves for each of the basic and tuned models.

Table I
SUMMARY OF PERFORMANCE METRICS

Model Name	Hyper-parameters	Metrics		
		Precision	*Recall*	*F_2-score*
Logistic Regression Classifier	*Untuned*	0.68	0.39	0.42
	Tuned	0.28	0.82	0.59
Random Forest Classifier	*Untuned*	0.43	0.43	0.43
	Tuned	0.31	0.75	0.59
Support Vector Classifier	*Untuned*	0.91	0.28	0.33
	Tuned	0.33	0.83	0.64

The ROC curves, depicting the level of congruency between true and false-positive rates for varying probability thresholds, indicate an improvement in each models separability following hyperparameter tuning, with the greatest increase seen in the RFC. Moreover, for greater values of 'recall', each model saw larger values of precision than before - this is most notable in the RFC and SVC. Notably though, the LR model, despite the clearly improved recall score shown in table I, changed the least according to figure 7. This may be due to its parameters being less critical than with RFC and SVC.

Achieving the highest F_2-score, the tuned SVC outperforms the other models, affirmed also by the greater ROC-AUC value and the attainment of higher 'precision' for better 'recall' values exemplified by the Precision-Recall curve. This performance does come at the cost of run-time: SVC boasts a large train-time complexity of $O(n^3_{\text{samples}} \cdot n_{\text{features}})$, resulting in a training time of nearly 5 minutes in view of the training set cardinality. A related limitation is that while LR and RFC were able to be tuned via the GridSearchCV feature of Scikit-learn, exhaustively assessing all combinations of specified parameters, the time complexity of SVC did not allow for this, so the weaker RandomizedSearchCV alternative was used, implying the possibility that better parameters for SVC exist than those found.

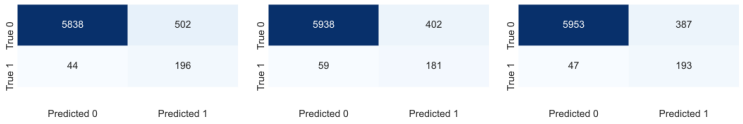

Figure 8. Confusion matrices on the testing data for the three tuned models. Left: LR; middle: RFC; right: SVC.

E. Feature importance

Finally, to assess the predictive values of the features, I utilised the technique of "permutation importance" from [9]. The model-agnostic technique consists of fitting the testing data to the model and making predictions several times, with the values of one feature shuffled at random prior to each. The *permutation importance* is then derived from the relative impact this has on the prediction. The results thereof are displayed in figure 9. The significance of the "age" feature is substantiated by all models. Of note is the otherwise unique array of importances attributed by each, suggesting that the different models ascertained distinct patterns in the data. Unsurprisingly, the SVC, which we earlier deemed to be the highest performing of the models, appears to have made greater use of the features.

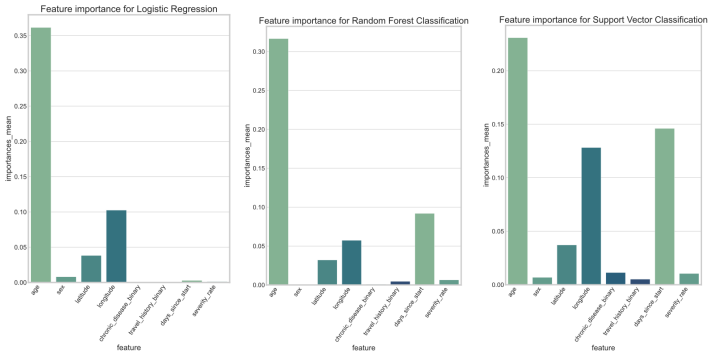

Figure 9. Mean permutation feature importances for each model.

IV. CONCLUDING REFLECTIONS

Many limitations were faced with respect to the models and the underlying dataset. Primarily, these limitations can be attributed to the data rather than the models. The original data came with vast quantities of missing values, with many potentially viable features such as "symptoms" being discarded. In addition, the substantial imbalance in outcomes, which has been discussed at length, required many mitigation measures.

The highest performing model was the SVC, which ultimately attained a recall score of 0.83 on the positive class, in spite of low precision. More intuition into its performance is available in figure 8, displaying its confusion matrix, from which it can be acknowledged that high recall and precision is attained with respect to the negative class. Semantically, the model accurately detects 83% of the severe cases - with a precision of one third - and 94% of the safe cases.

To attempt to improve the model further, in the future I would augment the dataset to increase the information associated with certain features. For instance, the geographic features could be used to obtain corresponding climate or economic information (e.g., the GDP per capita could be acquired for the "cities" feature), which may enable the models to discover more underlying patterns than those uncovered by these implementations.

REFERENCES

[1] B. Xu, B. Gutierrez, S. Mekaru, et al. (2020, Mar.). Epidemiological data from the COVID-19 outbreak, real-time case information. *Sci Data* [Online]. 7(106). Available: https://doi.org/10.1038/s41597-020-0448-0

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, et al. (2011, Dec.). Scikit-learn: machine learning in python. *Journal of Machine Learning Research* [Online]. 12(85), pp. 2825-2830. Available: https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

[3] Overfitting. (n.d.). In Oxford English Dictionary. Retrieved April 25, 2021, from https://www.lexico.com/definition/overfitting

[4] M. Ester, H. Kriegel, J. Sander, X. Xu. "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining*, Portland, OR, USA, 1996, pp. 226–231.

[5] N. Chawla, K. Bowyer, L. Hall, W. Kegelmeyer. (2002, Jun.). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research* [Online]. 16(1), pp. 321–357. Available: https://doi.org/10.1613/jair.953

[6] H. Han, W. Y. Wang, B. H. Mao., "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Proc. 2005 Int. Conf. Advances in Intelligent Computing*, Hefei, AH, China, pp. 878–887.

[7] G. Lemaître, F. Nogueira, C. K. Aridas. (2017, Dec.). Imbalanced-learn: a Python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research* [Online]. 18(17), pp. 1-5. Available: https://jmlr.csail.mit.edu/papers/v18/16-365.html

[8] T. Saito, M. Rehmsmeier. (2015, Mar.). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS One* [Online]. 10(3). Available: https://doi.org/10.1371/journal.pone.0118432

[9] L. Breiman. (2001, Oct.). Random Forests. *Machine Learning* [Online]. 45(1), pp. 5–32. Available: https://doi.org/10.1023/A:1010933404324