STA 160: CDC Final Report

CDC App Documentation: Time Series and Classification Methods

Ricardo Simpao, Falak Shah

Student ID: 914767659 Email: rlsimpao@ucdavis.edu
Student ID: 914663151 Email: fdshah@ucdavis.edu

Dr. Fushing Hsieh, Instructor
STA 160
University of California, Davis
June 9, 2021

STA 160 Project - Coronavirus Diagnosis App

1 Abstract

Since 2019, the coronavirus has engendered loss and confusion throughout the world, and even as COVID cases drop down in the United States in 2021, places such as India still suffer from coronavirus and its variant. For many people, they continue to experience constant stress from the uncertainty of the future, and particularly in countries still facing the coronavirus, medical professionals face the difficult decision on whether to allow patients use of hospital facilities, ventilators, or the intensive care unit (ICU). Our paper explores time series methodologies that would help forecast the current trend of cases in the United States, which we hope to expand to other countries and COVID variants. As of May 11, 2021, our paper forecasts an exponential decrease in COVID-19 cases in the U.S. throughout the summer. Furthermore, our paper finds logistic regression and support vector machines as the best models in classifying whether a particular patient would need hospitalization, ventilation, the ICU, and potentially, end-of-life care. Holistically, our paper works as the back-end documentation, particularly on the classification methods, for our accompanied Shiny app, which we made available to the public (https://rlsimpao.shinyapps.io/cdc-final/?_ga=2.22230386.897110583.1623115674-205289261.1623115674).

2 Introduction

The coronavirus has brought on a world-wide pandemic, disrupting the lives and livelihoods of millions of people. Symptoms, according to the World Health Organization website, generally appear as moderate fevers, dry coughing, and fatigue, while more serious manifestations appear as shortness of breath, chest pain, and loss of speech. While most people will tend to recover without need for hospitalization, the elderly, particularly those with compromised immune symptoms, may suffer more serious and life-threatening experiences. In countries where COVID-19 remains rampant, many families and doctors face the critical decision of how to take care of those afflicted with COVID. To aid in the decision making process, we evaluated different classification models, such as logistic regression, SVM, and random forest, to see which best classifies whether a patient will need hospitalization, ventilation, ICU, and end-of-life care. In addition, we built an accompanying Shiny application with the best classification models for each event. We hope that this may aid the decision making of hospitals with regards to allocation of their resources as well as in tracking the progression of the health status of individuals afflicted with coronavirus. Lastly, our paper attempts to assuage the fears and uncertainty of COVID-19 by projecting the future paths of COVID-19 cases as we enter through the summer of 2021 through univariate time series analysis, using the sarima model.

3 Related Works

Our report aims to replicate the ideas presented in the paper,“Personalized Predictive Models for Symptomatic COVID-19 Patients Using Basic Preconditions” by Wollenstein-Betech, Cassandras, and Paschalidis. The paper aims to predict the following events: hospitalization, mortality, need for ICU, and need for ventilation. They use data from 91,000 patients in Mexico with information on their demographics, medical conditions, and their given treatments. They leverage their data through different classification methods: logistic regression, support vector machines, random forests, and gradient boosted decision trees. Their models reached accuracies of 61%, 76%, 83%, and 84% for hospitalization, mortality, need for ICU, and need for ventilator, respectively. We wish to replicate and extend their findings to patients in the United States.

4 Methodology

4.1 Dataset

Our paper uses COVID-19 case data for patients in the United States, which dates from April 2020 to May 2021, from the Center for Disease Control and Prevention (CDC) website. The data contains roughly 260 million observations with information on patient's demographics, location, symptoms, survival, and whether or not they needed hospitalization, use of a ventilator, and use of the intensive care unit. Furthermore, for purposes of our classification, we used a random sample of the data set to alleviate problems of time complexity, associated with the data size. For our time series analysis, we used the entire data set and derived the counts of positive COVID-19 cases per day. The data contains only part of the entire population of cases, so for our time series analysis, we only look towards the trends of our analysis instead of the actual magnitudes of the forecasts. As such, we expect some underestimation in our forecasts.

4.2 Binary Logistic Regression

We aim to create binary classification models for each of the four events: hospitalisation, ICU need, Ventilator need, and Death. Binary logistic regression models the probability of each event, given the patient's symptoms and other patient information. The regression follows this form:

$$l = log(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

We use binary logistic regression to determine the relevance of multiple independent variables in predicting the classes (Hospitalisation, ICU, Ventilator and Mortality). Logistic regression forms the best fitting equation using maximum likelihood estimation (MLE). This maximises the probability of classifying the observed data into its appropriate category, given the regression coefficients of the independent variables. In our case, we implement logistic regression by modelling the posterior probability of target outcomes as a logistic equation of the linear combinations of our independent feature variables (Symptoms, Gender, Age Group and status of Covid test result).

Since logistic regression makes no assumptions about the distribution of classes in the feature space, and given its ease and efficiency in training and interpretation, we use it as our baseline classification model.

4.3 Support Vector Machine (SVM)

Support vector machines work by searching for a hyperplane in N-dimensional space, where N represents the number of features, that distinctly classifies the data points. Many possible hyperplanes exist that can separate the two classes of a given target variable, so we aim to find a plane that gives the maximum margin, or the maximum distance between data points of both classes. Maximising the margin distance provides some reinforcement that enables us to classify future data points with more confidence. Hyperplanes act as decision boundaries to help classify data points, and points on either side of the hyperplane are assigned to different classes. Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. These support vectors help build the SVM classifier and maximize its margin.

Here we compare the performances of binary logistic regression and SVM. In logistic regression, we take the output of the linear function and restrict the values within the range [0,1] using the sigmoid function. Whereas in SVM, the threshold values are changed to -1 and 1. Thus, the range [-1, 1] acts as the margin.

4.4 Random Forest

Random Forest is a supervised machine learning algorithm that uses ensemble learning for classification. Ensemble learning is a method that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. Its ability to work effectively with large data sets and handle many input variables makes it useful in classifying our four target variables. The Random Forest classifier is a set of decision trees from randomly selected subsets of the training set. It aggregates the votes from different decision trees to decide the final class of the test object. Each individual decision tree in the Random Forest generates a class prediction and the class with the most votes becomes the model's prediction. Random Forests use bagging and feature randomness to ensure model diversity. Decision trees are very sensitive to the data they are trained on, and takes advantage of this by allowing each individual tree to randomly sample from the dataset with replacement, resulting in different trees. This process is known as Bagging. Furthermore, when it is time to split a node, each tree in a random forest can pick from a random subset of features, allowing more variation and lower correlation across trees.

4.5 SARIMA

The sarima model is a time series model that incorporates seasonality, autoregressive models, and moving average models to fit time-dependent univariate data. Given some non-stationary data, the sarima model first differences the data, which basically subtracts more current data with less current data, to attain stationarity, and in addition, it uses seasonal differencing to remove season components. Once attaining the ideal stationary characteristics of time series, we can fit a sarima model to gain insight into the data's trends as well as forecast future values. The autoregressive aspect of the sarima model shows the relationship of the values based on its prior values, while the moving average model represents how the regression errors follows a linear combination of previous error terms.

5 Implementation Details

5.1 Preprocessing for Classification

5.1.1 Removing Missing and NA Values

In order to prepare the dataset for predictive modelling, we eliminate all the rows with missing or unknown values, consequently leaving us with 218,000 observations. Filtering also helps make our data set compatible for binary classification for all the four classification variables under consideration (Need for hospitalisation, ICU, Ventilator and Death). For purposes of classification analysis, we use variables, pertaining to the patient's symptoms, gender, age group, and the status of their Covid-19 test (Positive or Pending [sample still being tested in the laboratory]).

5.1.2 One-Hot Encoding

Since most of the machine learning algorithms cannot directly work with the data set's variable type, we need to employ one-hot encoding. One-hot encoding returns a column for each unique value of a given categorical column. For instance, if the column consists of a whether a patient had any pre-existing medical conditions, then one hot encoding creates two columns - "pre-existing medical conditions.yes" and "pre-existing medical conditions.no." Each of these columns is filled in by "0" or "1" depending on if the patient has any pre-existing medical conditions. The encoding helps us distinguish between the different categories of a feature. Since the levels of our categorical variables do not place importance in order, we opt to use one-hot encoding to prepare our data for the supervised machine learning algorithms.

5.1.3 Sampling for Class Imbalances: ROSE

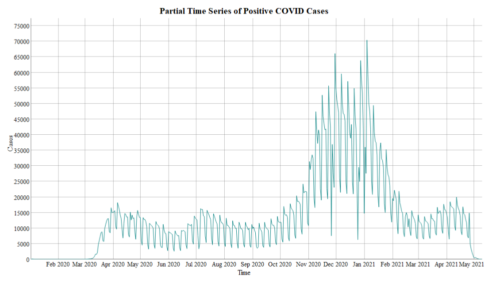
The preprocessed dataset is highly unbalanced and skewed towards the "No" class for each of our target variables. **Hospitalisation** has only 18609 "Yes" values (8.5 % of the total). **ICU** has 4774 (2.1 %) values, **Death** has 3775 (1.7 %) values and **Ventilator** has 2052 values(0.9 %). The imbalance leads to biased predictions and misleading classification accuracy. Given the size of the dataset, the algorithms used, and the time complexity, we use 10 % random sample of the data to build our model.

Furthermore, to deal with the imbalance in classes, we use the ROSE (Random OverSampling Examples) package in R to generate artificial data based on sampling methods and smoothed bootstrap approach for the minority class, which in our case is “Yes” for all the target variables. According to the package documentation, the ROSE function, a bootstrap-based technique, generates balanced samples and strengthens the subsequent estimation of any binary classifier. It handles both continuous and categorical data by generating examples from a conditional density estimate of the two classes.

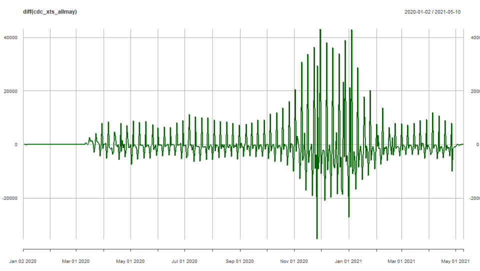
We also considered other methods to deal with class imbalance, such as oversampling the minority class and undersampling the majority class. However, data generated from undersampling may leave out important information from the original data. The ROSE function deploys a hybrid method, which counters these issues, so we opt to use the ROSE approach.

5.2 Pre-processing for Time Series

In order to obtain our time series, we simply group by each listed date (pos_spec_date) in the data set, ranging from around April 2020 to May 2021, and count all the listed observations for each date. We also remove any rows with empty dates. To evaluate the forecasting strength of our model, we also subset the data from March to October 31st as our validation set and compare our forecast data with the actual data. Furthermore, in order to fit our sarima model, we first need to conduct some differencing to make our data stationary, which we see in the plots below.



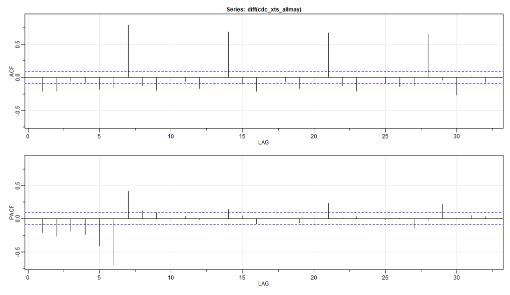
(a) Time Series Data



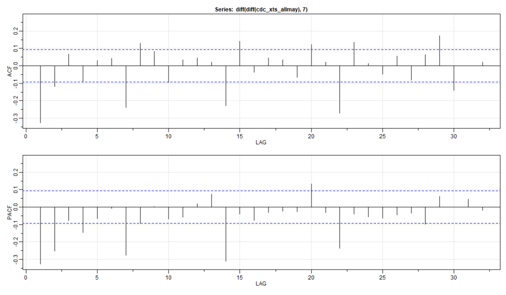
(b) Time Series Data with Diff = 1

5.3 Time Series: Visual Analysis

In order to determine the parameters for our sarima model, we need to look at the ACF and PACF plots of our stationary time series. With a differencing of 1, we see that there is a slow residual decay in the ACF plot at a seasonal lag period of 7 from Figure (a) below, which suggests we need to account for the seasonal lag. We see the lag occurs at periods of 7, so we tune the seasonal differencing parameter to a lag of seven, and we attain the ACF and PACF plots in Figure (b).



(a) ACF and PACF plot with Diff = 1



(b) ACF and PACF plot with Diff = 1, Seasonal lag = 7

To find the parameters for our sarima model, we observe the lags in the ACF and PACF plots in Figure (b). From the seasonal lag perspective, we can see that the ACF and PACF plots cut off after the second seasonal lag. This would suggest a SARIMA model of (2,2). Within the first seasonal cycle, both ACF and PACF seem to be cutting off after lag 2. Thus, our final proposed SARIMA model is **(2,1,2,2,1,2)[7]**, where 7 is the seasonal lag.

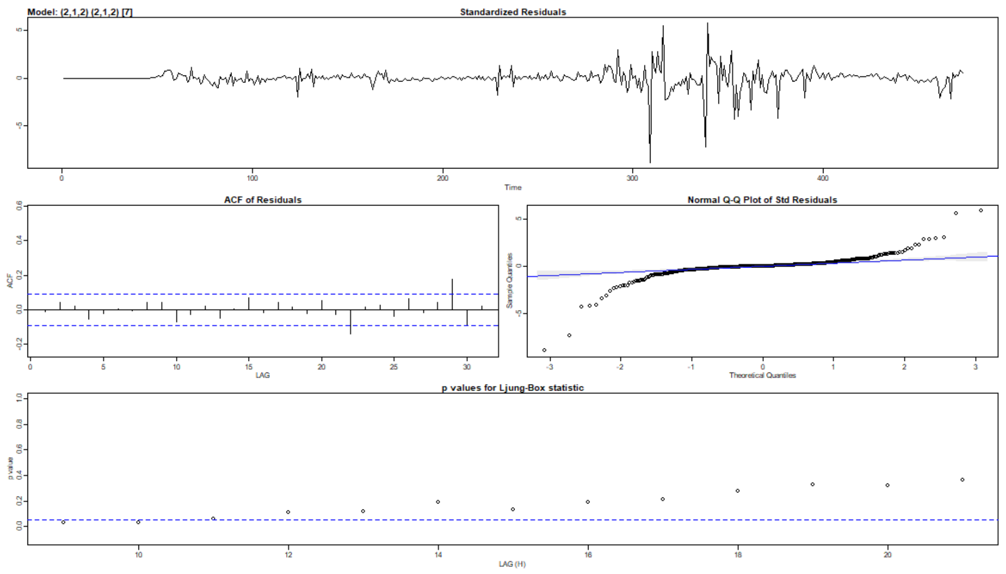
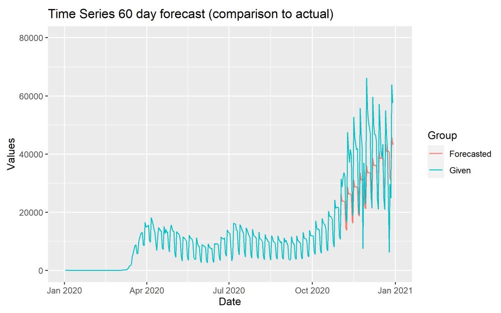
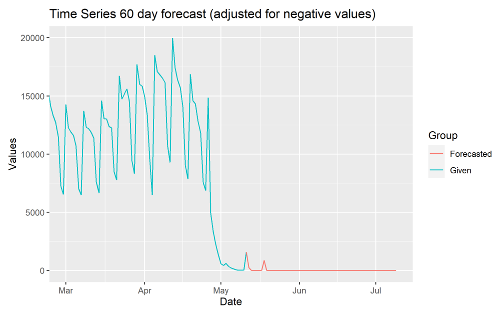


Figure 3: Diagnostics for sarima(2, 1, 2)(2, 1, 2)_7

To evaluate the performance of our sarima model, we look at the residual characteristics. From the diagnostic plots, we see that the model fits well, although there might still be some outliers in the data as shown by the QQ plot. Holistically, the residuals roughly satisfy normality. Furthermore, to evaluate our model's forecasting accuracy, we compare our forecast to actual values.



(a) Forecast 60 days after October 31, 2020



(b) Forecast 60 days after May 11, 2021

From our graph forecasting the COVID-19 cases from November to January 2021 (left), we see that the forecast correctly shows the upward trend in positive cases of coronavirus in the U.S. as it enters the winter months although it does not depict its high variance. We have reason to believe that our model reasonably forecasts the future cases of COVID-19 in the United States. From our graph for future cases from May 2021 and onward (right), we see a rapid exponential decline in our data, which correspondingly justifies the reopening of businesses, schools, and congregations of people in the United States. One caveat arises from our problems of forecasting negative cases in our time series despite our approaches of transforming our data. Negative cases of coronavirus do not make sense in context, so for simplicity of our analysis, we simply treat any negative forecasts as zero cases of COVID. Again, we note that we only use the partial dataset for our time series modelling and expect some underestimation in our results.

5.4 Predictive Modelling for the Diagnosis App

Procedure for Training Models:

To build personalised predictive models for each of our four target variables, we use the 'Caret' R package, which offers a comprehensive framework for building machine learning models as well as time-efficient procedures for finding the optimal model.

First, we sample 10% of our entire data for modelling and split that subset into training and test sets. We make a 75-25 split, with majority as the training set, using the createDataPartition() function, which preserves the proportion of the categories in our target variable. We then use the ROSE function on the training set to balance the distribution of our classes - whether or not a certain event occurs - to build more robust predictive models. After applying this technique, for each of our target variables, we have about 8200 0's ("No" values) and 8100 1's ("Yes" values) in our training dataset. We apply the ROSE function only to the training set and leave the test set unchanged to avoid possible overfitting in our model.

The train function helps us cross-validate the model, tune the hyperparameters for optimal model performance, and choose the optimal model based on a variety of optional evaluation metrics. We employ a 5-fold cross validation for training all the different models for our target variables. In other words, we use bootstrap resampling and collect the evaluation metrics for each bootstrap sampling. Ultimately, we can create a grid of parameters for model training, and then train the model on slightly different data (due to cross validation) for each candidate combination of tuning parameters. Across each dataset, we calculate the performance of the held-out samples and summarize evaluation metrics for each combination. We then choose the combination with the ideal resampling statistic as the final model, and we use the entire training set to fit the final model.

5.4.1 Hyperparameter tuning

In terms of regularization (hyperparameter tuning) for logistic regression, the train function helps tune the alpha and lambda parameters. Alpha is the parameter that adds a penalty based on the number of features to prevent overfitting. Lambda controls the leaning rate (how much the model changes during each iteration of learning). For Linear SVM, the parameter "C" is tuned. It controls how much you want to avoid misclassifying training samples. Alternatively, it determines how hard or soft the hyperplane margin classifier should be. For Random forest, we tune the "mtry" parameter - the number of variables to randomly sample at each split.

5.4.2 Model Evaluation

Lastly, to select the optimal training model for each of the three algorithms, we use **ROC (Receiver Operating Characteristic)** as the evaluation metric instead of accuracy. Accuracy is based on one specific cutpoint, whereas ROC tries all the different cutpoints and plots the sensitivity and the specificity. Sensitivity is the proportion with which a classifier correctly predicts a positive result. In our case, the positive class is "No", whereas specificity is the proportion with which a classifier correctly predicts the negative result, which in our case is "Yes". Although we balance the distribution of classes in our training set, to deal with any potential remaining skewness, we use ROC because it accounts for sensitivity and specificity. A higher accuracy score could be misleading as it tends to predict the dominant class well and the minority class poorly. Hence, we use ROC, and choose the combination of hyperparameters with the highest ROC value as the optimal training model for each algorithm.

We perform the mentioned process for training all the models for each of our four target variables. The metrics used to evaluate the performance of the test set are explained in the "Results" section.

5.4.3 Hospitalization

The model for hospitalization predicts whether a patient, who has tested positive for COVID- 19 or is waiting for the test result, would need hospitalization, given their conditions (symptoms, age group, gender). The training models selected for logistic regression, SVM and Random Forest can be observed in the figures below. The figures (a), (b) and (c) show the output of the trained models and the regularization of parameters. The optimal trained logistic model has **alpha = 0.10** and **lambda = 0.0169**. We choose this as it has the highest ROC value of **0.8708**. Similarly, the optimal SVM model has **C = 1** and **ROC = 0.8658**; the optimal Random Forest model has **mtry = 7** and **ROC = 0.99**.

In terms of the performance of these 3 models on the test data set, all the models perform decently with logistic regression having an “Area Under the Curve” of 0.843, SVM having an AUC of 0.836 and Random Forest having an AUC of 0.7631. We discuss the performance evaluation further in the Results section.

The table below shows the 10 most important variables for predicting hospitalization for each of the three methods. As we can observe, age dominates the importance for logistic regression. We see similarities in the importance of a few symptoms for all the three methods such as “Abnormal Chest X-Ray”, “Pneumonia”, “Acute Respiratory Distress Syndrome [ARDS]” and “Shortness of breath”, although the magnitude of importance varies amongst the methods.

Note: The values inside “[]” depict the magnitude of importance of a particular feature on a scale of 0-100

Variable Importance			
	Logistic Regression	**SVM**	**Random Forest**
1	Age(80+)[100]	Abnormal Chest X-Ray[100]	Abnormal Chest Xray[100]
2	Abnormal Chest X-Ray[87.72]	Pneumonia[94.79]	ARDS [94.43]
3	Pneumonia[87.30]	Shortness of Breath[77.06]	Pneumonia[75.39]
4	Age(70-79)[85.3]	Pre-Existing Medical Condition[70.99]	Age(80+)[19.64]
5	Age(0-9)[79.62]	Fever above 100.4F[42.29]	Age Group(10-19)[17.97]
6	Age(20-29)[75.88]	ARDS[33.88]	Age(70-79)[10.92]
7	Age(10-19)[69.01]	Headache[26.49]	Age Group(20-29)[10.08]
8	Shortness of Breath[53.81]	Cough[26.2]	Age Group(0-9)[7.63]
9	ARDS [45.57]	Runny nose[24.98]	Shortness of Breath[4.37]
10	Pre Existing MC[44.99]	Age(80+)[23.42]	Pre Existing MC[4.22]

Table 1: *10 most important variables to predict Hospitalization for each algorithm*

5.4.4 ICU

Similar to hospitalization, the model for ICU predicts whether a patient, who has tested positive for COVID-19 or is waiting for the test result, will need an ICU (Intensive Care Unit) given their conditions. The training models for the three methodologies are chosen exactly like the corresponding ones for hospitalizations.The optimal trained logistic model has **alpha = 0.10**,**lambda = 0.0005** and **ROC = 0.937**. Similarly, the optimal SVM model has **C = 1** and **ROC = 0.931**; the optimal Random Forest model has **mtry = 7** and **ROC = 0.998**.

In terms of the performance of these 3 models on the test data set, all the models perform well with logistic regression with an AUC of 0.8891, SVM with an AUC of 0.8807 and Random Forest with an AUC of 0.8707.

The table below shows the 10 most important variables for predicting the need for ICU for each of the three methods. Apart from the variables mentioned in the table, the result of the COVID-19 test and whether the person is a health worker, are important variables for Random Forest, although they have a smaller impact on the overall outcome. Similarly for SVM and logistic regression, the status of the COVID test and gender have a smaller impact on the outcome.

Note: The values inside "[]" depict the magnitude of importance of a particular feature on a scale of 0-100

	Variable Importance		
	Logistic Regression	**SVM**	**Random Forest**
1	Age(20-29)[89.51]	Abnormal Chest X-Ray[99.81]	Abnormal Chest Xray[93.14]
2	Abnormal Chest X-Ray[87.89]	Pneumonia[91.72]	ARDS[92.57]
3	Age(10-19)[85.79]	Shortness of Breath[91.72]	Pneumonia[68.20]
4	ARDS[79.57]	Pre-Existing MC[60.52]	Age(10-19)[24.65]
5	Age(70-79)[66.32]	ARDS[59.28]	Pre existing MC[19.03]
6	Pneumonia[54.05]	Fever above 100.4F[40.46]	Age(20-29)[18.57]
7	Pre existing MC[51.39]	Headache[24.88]	Age Group(0-9)[13.75]
8	Shortness of Breath[50.21]	Cough[24.55]	Age Group(80+)[8.42]
9	Age(30-39)[45.57]	Male[23.08]	Age(70-79)[8.35]
10	Age(60-69)[44.67]	Age(60-69)[21.58]	Shortness of Breath[6.89]

Table 2: *10 most important variables to predict ICU need for each algorithm*

5.4.5 Mortality

The model for Mortality predicts whether a patient, who has tested positive for COVID- 19 or is waiting for the test result, will stay alive given their conditions. The optimal trained logistic model has **alpha = 0.10**,**lambda = 0.013** and **ROC = 0.933**. Similarly, the optimal SVM model has **C = 1** and **ROC = 0.932**; the optimal Random Forest model has **mtry = 7** and **ROC = 0.98**.

In terms of the performance of these 3 models on the test data set, all the models perform reasonably with logistic regression having an AUC of 0.8868, SVM having an AUC of 0.8957 and Random Forest having an AUC of 0.8924.

The table below shows the 10 most important variables for predicting mortality for each of the three methods. As we can observe, apart from the usual features like "Age", "Abnormal Chest X-Ray", "Pneumonia", "ARDS", and "Pre-Existing Medical Condition", features like headache, runny nose, sore throat, muscle ache and being a health worker play a role, although a small one, in determining mortality in contrast to determining the need for ICU or hospitalization.

Note: The values inside "[]" depict the magnitude of importance of a particular feature on a scale of 0-100

	Variable Importance		
	Logistic Regression	**SVM**	**Random Forest**
1	Age(80+)[100]	Pre Existing MC[99.76]	Age(80+)[100]
2	Age(70-79)[85.59]	Age(80+)[95.28]	ARDS[80.95]
3	Age(30-39)[77.29]	Abnormal Chest X-Ray[94.85]	Age (20-29)[56.27]
4	ARDS[41.69]	Pneumonia[94.15]	Health Worker[40.52]
5	Health Worker[41.06]	Headache[88.57]	Abnormal Chest X-Ry[34.96]
6	Pre Existing MC[34.76]	ARDS[72.03]	Pneumonia[30.77]
7	Abnormal Chest X-Ray[30.10]	Pre Existing MC[14,47]	Age Group(20-29)[10.08]
8	Headache[28.06]	Muscle Ache[61.13]	Age(70-79)[7.71]
9	Pneumonia[22.21]	Runny nose[38.05]	Headache[4.76]
10	Muscle Ache[21.85]	Sore throat[37.78]	Muscle Ache[1.17]

Table 3: *10 most important variables to predict Mortality for each algorithm*

5.4.6 Mortality (second model including status of hospitalization, ICU and need for ventilator)

We fit a second model for Mortality, which includes the status of hospitalization, need for ICU, and Ventilator as well. We use the first model to predict mortality based solely on the base features mentioned prior to hospitalization, whereas the second model predicts mortality of a patient given they have been hospitalized and/or needs an ICU and/or a ventilator. This model helps track the progress of the disease.

The AUC for the optimal trained models for logistic regression, SVM and Random Forest for the second model is 0.9183, 0.9186 and 0.9048 respectively. These values are all better than their corresponding values for the first Mortality model. Hence, it suggests that whether a person is hospitalized or not is an important feature when predicting mortality, justified by chart of variable importance which places hospitalization as the 5th most important variable for logistic regression with an importance score of 44.15. Similarly, for SVM and Random Forest, hospitalization is the 1st and 3rd most important variable with importance scores of 100 and 63.43 respectively. Ventilator need is the 6th most important variable for logistic regression (score = 40.44); ICU need is the 5th most important variable for SVM (score = 56.39). Finally, for Random Forest, need for Ventilator is the 1st most important variable (score = 100) and need for ICU is the 4th most important variable (score = 31.09).

5.4.7 Ventilator

The model for Ventilator predicts whether a patient, who has tested positive for COVID- 19 or is waiting for the test result, will need a ventilator given their conditions. The optimal trained logistic model has **alpha = 0.10**,**lambda = 0.0004** and **ROC = 0.951**. Similarly, the optimal SVM model has **C = 1** and **ROC = 0.945**; the optimal Random Forest model has **mtry = 7** and **ROC = 0.99**.

In terms of the performance of these 3 models on the test data set, logistic regression and SVM perform well but Random Forest performs poorly. The results follow as: logistic regression with an AUC of 0.9044, SVM with an AUC of 0.9031 and Random Forest with an AUC of 0.5352.

The table below shows the 10 most important variables for predicting the need for ventilator for each of the three methods. As we can observe, apart from the usual features mentioned above, symptoms like abdominal pain, fever (above 100F), status of COVID-19 test result and gender play a more prominent role in determining ventilator need in comparison to predicting the other three target variables.

Note: The values inside "[]" depict the magnitude of importance of a particular feature on a scale of 0-100

	Variable Importance		
	Logistic Regression	**SVM**	**Random Forest**
1	ARDS[85.33]	Abnormal Chest X-Ray[99.24]	Abnormal Chest Xray[93.26]
2	Abnormal Chest X-Ray[83.09]	Pneumonia[96.34]	Confirmed COVID 19 case[86.21]
3	Covid Confirmed case[67.48]	ARDS[79.99]	Probable COVID-19 case[84.09]
4	Pneumonia[61.77]	Shortness of Breath[67.61]	ARDS[75.78]
5	Covid Probable case[53.58]	Pre Existing MC[51.82]	Pneumonia[67.16]
6	Age(70-79)[38.62]	Fever above 100F[38.59]	Shortness of Breath[8.603]
7	Age(80+)[37.07]	Headache[20.72]	Pre existing MC[6.89]
8	Pre existing MC[35.83]	Cough[20.10]	Age Group(80+)[4.48]
9	Abdominal Pain[34.01]	Male[19.09]	Age Group(70-79)[3.55]
10	Shortness of Breath[29.81]	Runny nose[18.23]	Health Worker[3.26]

Table 4: *10 most important variables to predict ventilator need for each algorithm*

5.4.8 Ventilator (second model - including status of ICU need)

Similar to our approach with Mortality, we consider a second model to classify the need for Ventilator by adding the status of ICU need as another feature.

The AUC for the optimal trained models for logistic Regression, SVM and Random Forest for the second model is 0.95, 0.953 and 0.5352 respectively. These values are at least equal to (for Random Forest) or better (for logistic regression, SVM) than their corresponding values for the first Mortality model. Hence, it suggests that ICU need could be a crucial feature to predict ventilator need. This is confirmed by the variable importance chart which places ICU need as the 1st most important variable for logistic regression with an importance score of 88.91. Similarly, for SVM and Random Forest, ICU need is the 1st most important variable with importance scores of 99.72 and 100 respectively.

6 Results

6.1 Evaluation Metrics

To evaluate the performance of our classifiers, we consider a few metrics, apart from accuracy, which help us capture the interpretability and accuracy of prediction for both classes, Yes and No.

To check the accuracy of our model for predicting both classes, we report the sensitivity and specificity scores for the test set of each model (Logistic Regression, SVM and Random Forest) for each of the four target variables (Hospitalization, ICU need, Ventilator need and Mortality). Sensitivity measures how often a test correctly generates a positive result for the people who have that positive condition, as defined by the problem. In other words, Sensitivity is known as the true positive rate and it is the ratio of $\frac{True\ positives}{TruePositives\ +\ False\ Negatives}$. It tells us about the proportion of actual positive cases that were predicted as positive by our model. In this case, for each of our target variable, the positive case is "No". For hospitalization, for instance, sensitivity gives the proportion of actual number of patients who were not hospitalised that were predicted as not hospitalized or "No" by our model. Similarly, specificity is known as the true negative rate. In our model, the negative class is "Yes". Hence, for hospitalization, specificity gives us the proportion of actual number of patients who were hospitalized that were predicted as hospitalized or "Yes" by our model. It is the ratio of $\frac{True\ Negatives}{True\ Negatives\ +\ False\ Positives}$

Apart from these two metrics, we also report the Area Under the Curve (AUC) of ROC (Receiver Operator Characteristic). The ROC curve, an evaluation metric for binary classification problems, is a probability curve that plots the True Positive Rate (Sensitivity) against the False Positive Rate (Specificity) at various threshold levels and separates the signal from the noise. The AUC provides an aggregate measure of performance across all possible classification thresholds and summarizes the ROC curve. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes. Hence we use AUC as our guiding metric to choose the best model instead of accuracy ($\frac{Number\ of\ correct\ predictions}{Total\ predictions}$) since a high accuracy score can be misleading if it predicts the majority class well and the minority class poorly.

6.2 Method Comparison/ Final Model selection

We can see the test set performance of all the three methods for the Hospitalization model below (Table 5). Although the accuracy of Random Forest is high, the specificity is considerably lower than the other two methods, which impacts its AUC Score as well. Random Forest does not do a good job of predicting the "Yes" rows correctly, but because it does a solid job in predicting the majority class in the test set, "No", the accuracy is high. The two methods, logistic regression and SVM, have a high score for all the metrics. The final model chosen for classifying Hospitalization is **Logistic Regression** because it has a higher AUC score than SVM, and it predicts "Yes" slightly better than SVM does (Specificity). Furthermore, because it is crucial to predict the "Yes" class, given our problem, we opt to use logistic regression over SVM even though the latter does slightly better on the sensitivity metric.

Test Set Performance Evaluation for Hospitalization					
	Method	Sensitivity	Specificity	AUC	Accuracy
1	Logistic Regression	0.8904	0.7956	0.843	0.8826
2	SVM	0.9031	0.7689	0.836	0.8913
3	Random Forest	0.9773	0.5489	0.7631	0.942

Table 5: *Test Set Performance Evaluation for Hospitalization*

We can see the test set performance of all the three methods for the ICU need model in Table 6 below. Here the performance of the three methods across all the metrics is closer than the model above, with all of them scoring highly across different parameters. As the final model for classifying ICU need, we choose **Logistic Regression** because of its higher AUC score than SVM and Random Forest.

Test Set Performance Evaluation for ICU Need					
	Method	Sensitivity	Specificity	AUC	Accuracy
1	Logistic Regression	0.9341	0.844	0.8891	0.9323
2	SVM	0.9446	0.8165	0.8805	0.942
3	Random Forest	0.9524	0.789	0.8701	0.9491

Table 6: *Test Set Performance Evaluation for ICU Need*

We can see the test set performance of all the three methods for Ventilator need (First Model) in Table 7 below. Here, Random Forest performs significantly worse compared to the two other methods. As the final model for classifying Ventilator need, we choose **Logistic Regression** because of the higher AUC score. The other models perform almost on par with logistic regression.

Test Set Performance Evaluation for Ventilator Need (First model)					
	Method	Sensitivity	Specificity	AUC	Accuracy
1	Logistic Regression	0.9516	0.8571	0.9044	0.9508
2	SVM	0.949	0.8571	0.9031	0.9482
3	Random Forest	0.07	1	0.5352	0.07

Table 7: *Test Set Performance Evaluation for Ventilator Need (First Model)*

However, given that we also account for ICU need, we choose **SVM** as the final model because of its higher AUC score. Both models perform on par with SVM faring minutely better on the Sensitivity and AUC score. (Table 8)

Test Set Performance Evaluation for Ventilator Need (Second model)					
	Method	Sensitivity	Specificity	AUC	Accuracy
1	Logistic Regression	0.9815	0.9184	0.95	0.981
2	SVM	0.9876	0.9184	0.953	0.987
3	Random Forest	0.07	1	0.5352	0.079

Table 8: *Test Set Performance Evaluation for Ventilator Need (Second Model)*

We can see the test set performance of all the three methods for classifying Mortality (First Model) in Table 9 below. Once again, the performance of all the methods remain close. We choose **SVM** as the final model chosen for classifying Mortality (First Model) because of its slightly higher AUC and specificity score than SVM and Random Forest.

Test Set Performance Evaluation for Mortality (First model)					
	Method	Sensitivity	Specificity	AUC	Accuracy
1	Logistic Regression	0.8932	0.8804	0.8868	0.893
2	SVM	0.9001	0.8913	0.8957	0.899
3	Random Forest	0.9153	0.8696	0.8924	0.9146

Table 9: *Test Set Performance Evaluation for Mortality (First Model)*

The performance of the methods improves in the second model once the status of Hospitalization, Ventilator and ICU need are included as features, although the pattern remains the same. We choose **SVM** as the final model for classifying Mortality (second model) because of its slightly higher AUC score than Logistic Regression and Random Forest (Table 10).

Test Set Performance Evaluation for Mortality (Second model)					
	Method	Sensitivity	Specificity	AUC	Accuracy
1	Logistic Regression	0.9283	0.913	0.9183	0.9233
2	SVM	0.9241	0.913	0.9186	0.9239
3	Random Forest	0.9401	0.8696	0.9048	0.9389

Table 10: *Test Set Performance Evaluation for Mortality (Second Model)*

7 Conclusion

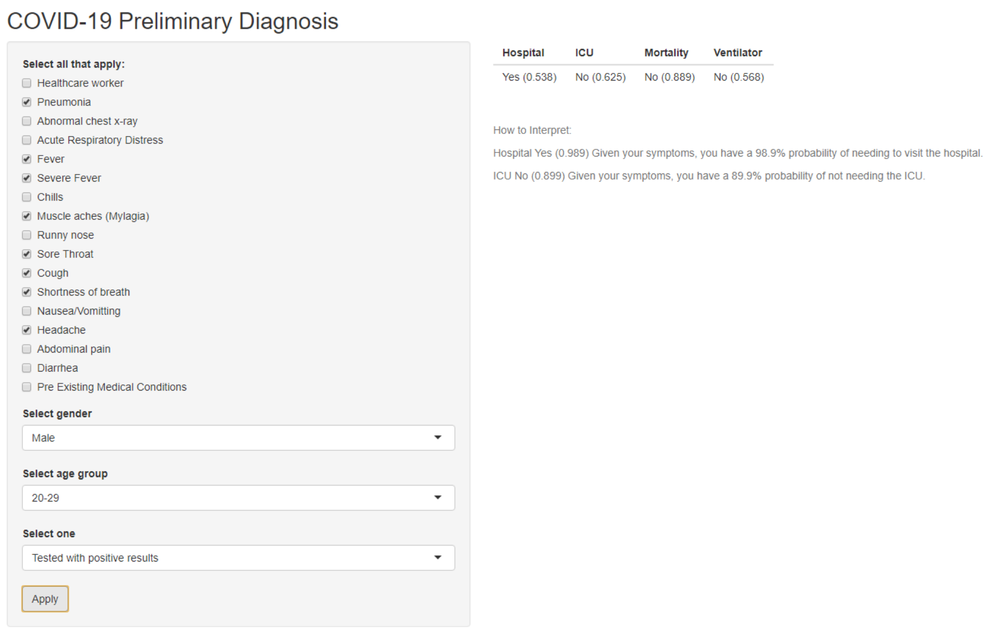
In forecasting the future trends of cases in the United States, we expect exponential decline in the next three months following May 11, 2021. Our findings justify the reopening of businesses and transition to less restrictive quarantine procedures as the United States enters the summer months. To expand our findings, we wish to apply our time series models towards countries where COVID-19 still poses as a viral threat to many people. Furthermore, we hope that it may also help in tracking COVID variant cases.

In terms of classification, overall, the performance of the two methods, logistic regression and SVM, proves better and more consistent than Random Forest for all the four target variables. Although Random Forest performs well for classifying Mortality and ICU need, its performance dips considerably while predicting the remaining two target variables - Ventilator Need and Hospitalization. Specifically, we find that logistic regression performs the best, across metrics, in determining whether a person would need hospitalization, ICU, or the ventilator given their specified conditions. For classifying for the need of end-of-life care, we use support vector machines as our model. We use these models as the classifiers for our accompanied diagnostic application. For the purpose of testing the improvements in our models, we fit additional models for classifying Ventilator need and Mortality by considering a few more features. Given the importance of allocating medical resources judiciously during this pandemic, we hope that our analysis and the accompanying application will aid the public and people working in the medical sector in making the best decisions for the people in their care.

8 Code Appendix

All available code can be found on the public Github repo https://github.com/dlsimpao/CDC-Coronavirus. The repository contains code on pre-processing, classification models, time series models, and Shiny app code.

8.1 Additional Material



COVID-19 Preliminary Diagnosis

Select all that apply:

- Healthcare worker
- [✓ Pneumonia]
- Abnormal chest x-ray
- Acute Respiratory Distress
- [✓ Fever]
- [✓ Severe Fever]
- Chills
- [✓ Muscle aches (Mylagia)]
- Runny nose
- [✓ Sore Throat]
- [✓ Cough]
- [✓ Shortness of breath]
- Nausea/Vomitting
- [✓ Headache]
- Abdominal pain
- Diarrhea
- Pre Existing Medical Conditions

Select gender: Male

Select age group: 20-29

Select one: Tested with positive results

Hospital	ICU	Mortality	Ventilator
Yes (0.538)	No (0.625)	No (0.889)	No (0.568)

How to Interpret:

Hospital Yes (0.989) Given your symptoms, you have a 98.9% probability of needing to visit the hospital.

ICU No (0.899) Given your symptoms, you have a 89.9% probability of not needing the ICU.

Figure 5: Screenshot of the diagnosis Shiny Application

The diagnosis app works by prompting the user to select the conditions, which apply to them. In the back end, this creates a row of observations with values for each variable given. Upon pressing the “Apply“ button, the app predicts the person’s need for hospitalization, ventilation, the ICU, and end-of-life care.

Features in the data set		
	Variable	**Feature**
1	current_status	Status of COVID (Positive or Probable)
2	sex	Gender
3	age_group	Age Group (0-9,10-19,20-29......70-79,80+)
4	hosp_yn	Was the patient hospitalized
5	icu_yn	Was the patient admitted to an Intensive Care Unit
6	mechvent_yn	Did the patient receive Mechanical Ventilation
7	death_yn	Did the patient die off this illness
8	hca_work_yn	Is the patient a health care worker in the US
9	pna_yn	Did the patient develop pneumonia
10	abxchest_yn	Did the patient have an abnormal chest X-Ray
11	acuterespdistress_yn	Did the patient have Acute Respiratory Distress Syndrome (ARDS)
12	fever_yn	Fever greater than 100.4F
13	sfever_yn	Subjective Fever (Felt feverish)
14	chills_yn	Chills
15	myalgia_yn	Muscle aches
16	runnose_yn	Runny nose
17	sthroat_yn	Sore throat
18	cough_yn	Cough
19	sob_yn	Shortness of breath
20	nauseavomit_yn	Nausea or Vomiting
21	headache_yn	Headache
22	abdom_yn	Abdominal pain
23	diarrhea_yn	Diarrhea
24	medcond_yn	Pre-Existing Medical conditions

Table 11: *Variables in the data set*

Table 11 above shows all the different features available in the datatset.
Note: Every symptom has two levels - Yes/ No.

9 Supplementary Material

The zip file contains app code, given in the global, server, and ui R files. Both 'cdc_allmay_positive_cases_ts.csv' and 'cdc_filtered_encoded.csv' represent our preprocessed data files we use for our analysis. The 'cdc_preprocessing.Rmd contains all the code for preprocessing our data for time series and classification. 'Classification - Final Project.Rmd' contains the classification code for our six models. 'cdc_time_series_model.Rmd' contains the code for our time series analysis.

10 References

1. Personalised Predictive Model for Covid-19: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7273257/
2. Handling Imbalanced Datasets for Classification: https://medium.com/@cmukesh8688/how-to-handle-imbalanced-classification-problems-4a96f42ae4c4
3. Machine Learning in R using CARET package: https://www.machinelearningplus.com/machine-learning/caret-package/
4. Train function documentation (CARET): https://rdrr.io/cran/caret/man/train.html
5. Understanding Sensitivity and Specificity: https://www.healthnewsreview.org/toolkit/tips-for-understanding-studies/understanding-medical-tests-sensitivity-specificity-and-positive-predictive-value/
6. Logistic Regression: https://bookdown.org/chua/ber642_advanced_regression/binary-logistic-regression.html
7. Advantages of Logistic Regression: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/

8. SVM Theory: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

9. Understanding Random Forest: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

10. Implementing ROSE in R: https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/ -

11. ROSE documentation: https://cran.r-project.org/web/packages/ROSE/ROSE.pdf

12. Influence of C in SVM: https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel

13. ROC vs Accuracy: https://stats.stackexchange.com/questions/68893/area-under-curve-of-roc-vs-overall-accuracy

14. Importance of ROC and AUC: https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/

15. SARIMA Modelling and Forecasting in R: https://medium.com/@kfoofw/seasonal-lags-sarima-model-fa671a858729

16. Time Series Cross Validation: https://robjhyndman.com/hyndsight/tscv/

17. Time Series Layout: https://www.kaggle.com/szrlee/time-series-analysis-arima-basic-model