Introduction: (proper formatting here)

The COVID-19 Time to Hospitalization datasets consist of a training set of 219 individuals and a testing set of 200 individuals; the datasets contain several columns related to COVID-19 cases including location data, sex, and date/time data. The goal of our analysis is to make a prediction on the column duration,the amount of time elapsed between onset of symptoms and hospitalization, with the lowest possible RMSE.

Our group, *kaggle username 6*, has attempted 6 different machine learning techniques and one classical statistical model in order to predict duration. The machine techniques are: *LASSO, Ridge, Random Forest, KNN* and *SVM regression* and a *Neural Network;* the classical statistical model is *gamma regression*. Preceding the models the data was cleaned, separating out symptoms as indicators and combining them into varying levels and imputing the missing values. Once the data was cleaned, which features could be dropped to minimize RMSE and optimization of parameters for each of the methods was explored.
We found that using optimal parameters and combinations of variables, almost every method achieved an RMSE between 4.7-4.5, however, using *Random Forest regression* we were able to achieve an RMSE below 4.5 and we have chosen this as our final model.

Data cleaning levels:

The age column contained both range and numeric values. We took the mean of each of the ranges in order to make this a completely numeric column clean_age.

Since duration and clean_age are the only continuous numeric columns, we made scatter plots of these two variables colouring by the categorical variable levels to make an initial assessment on the distribution . We detected no clear patterns(figure 1.)

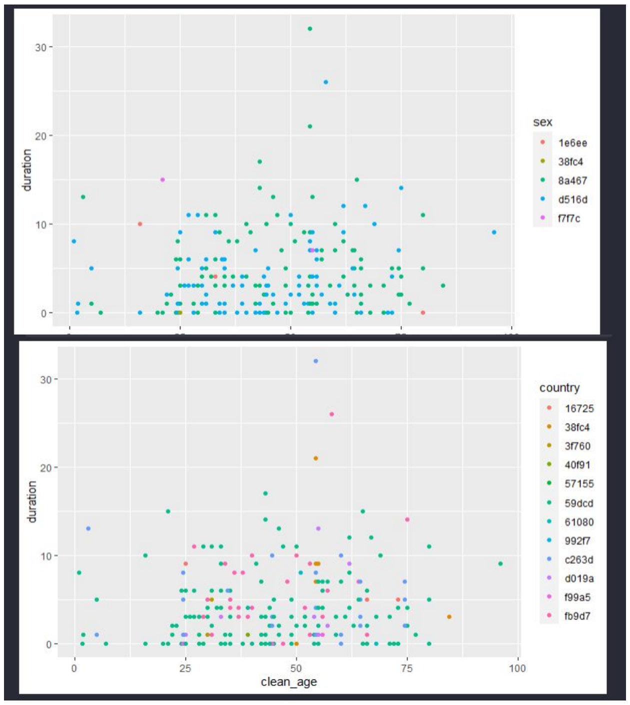


Figure 1. Scatter plots of `duration` vs `clean_age`and coloured by `sex` and `country`

Symptoms were organized as a comma separated list of all symptoms for a given individual, e.g. “cough, fever, sore throat” under the symptoms column. To make these features useful they were parsed into indicators (True, False) making a new column for each unique symptom. This was done with RegExs to attempt to capture symptoms outside of the training set.

Several of the symptoms were able to be grouped, we did this in three levels as follows:
Level 1: Grouping symptoms clearly identical such as ‘diarrhea’,‘diarrheoa’, and ‘shortness of breath’, ‘shortness breath’.
Level 2: Combining similar levels, eg. 37degree fevers into low fever and others fevers into high fever
Level 3: Combining all respiratory, fever, GI tract symptoms into their own columns.

For level 3 we also binned the clean_age (0-2,3-18,19-25,26-45,46-65,65+) on the intuition that very young children and the elderly may be admitted more quickly to the hospital. An attempt was made to bin the confirmed column, but this always gave us a much higher RMSE and was reverted.
The distribution of duration was plotted as a histogram with stacked colours by levels of the categorical variables to help us visualize variables that have a different ratio of levels for different lengths of duration. The intuition here is that the variables which have varying ratios with duration will be better predictors. In figure 2. one can see that sex does not appear to vary with disribution wheras clean_age, respiratory and VI do.

Interpolate NA with MICE:
We used the mice package in *R* to impute missing values in both train and test data. By

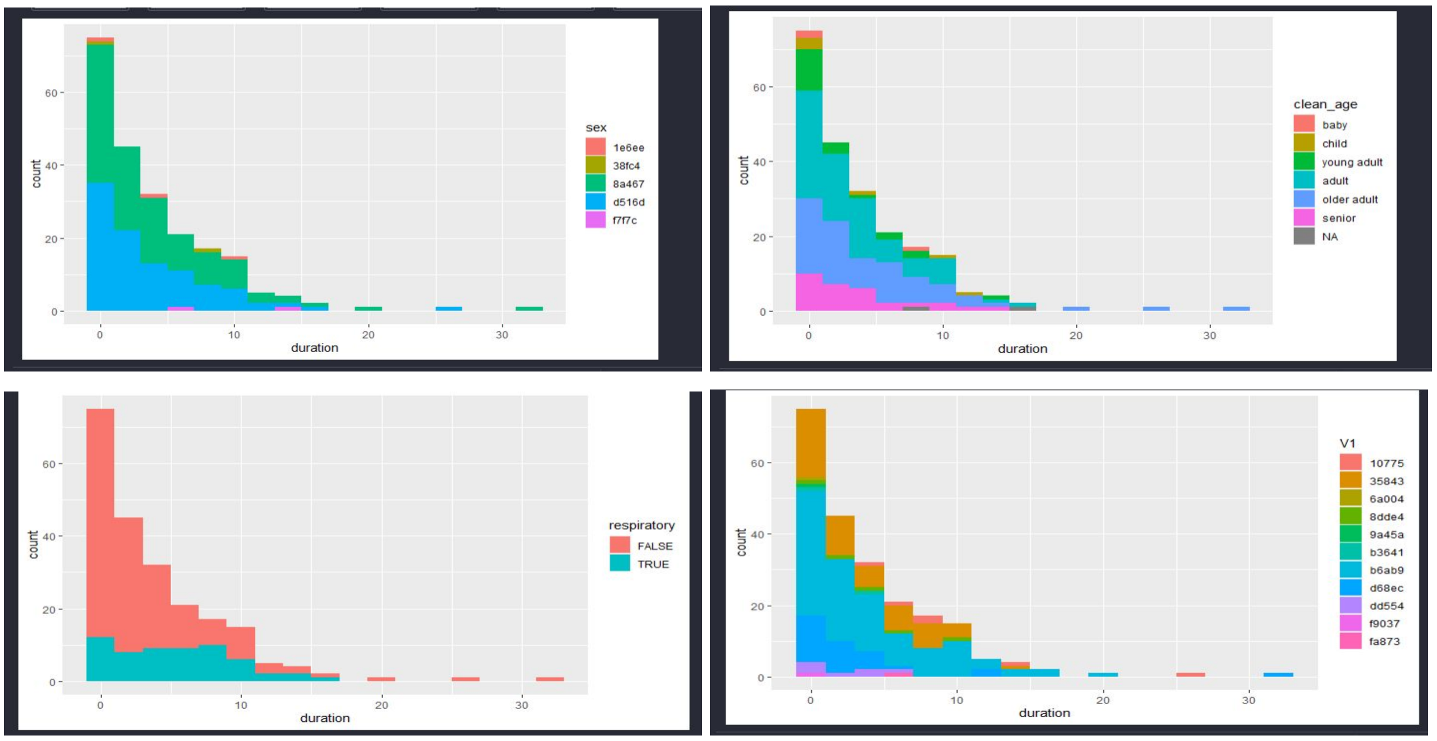


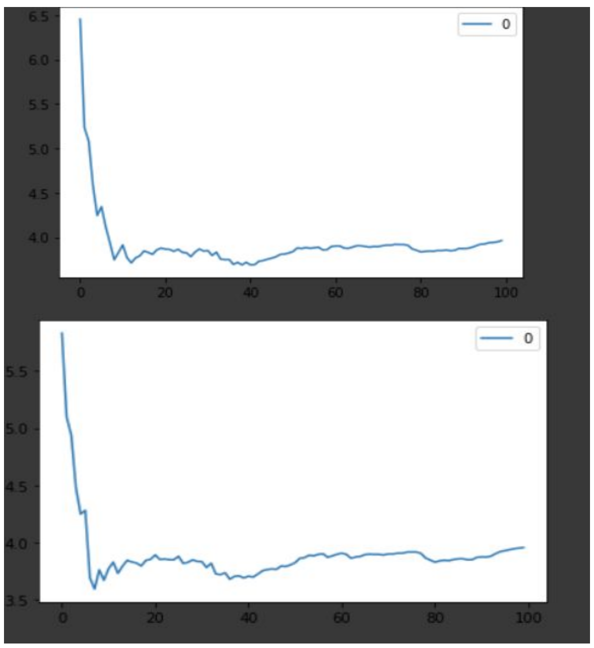
Figure 2. Histogram of `duration` colour stacked with `clean_age`, `sex`, `respiratory`, and `V1`

calculating percentages of missing values of each column, we can see in the output below there are missing data in clean_age (0.91%) and confirmed (0.46%)

```
       x              age             sex            city
0.000000         0.000000        0.000000        0.000000
province          country              v1       confirmed
0.000000         0.000000        0.000000        0.456621
duration        clean_age          nausea        vomiting
0.000000         0.913242        0.000000        0.000000
```

After applying the *Predictive Mean Matching* method for continuous clean_age and *Bayesian Polytomous Regression* for categorical clean_age and confirmed, the percentages of missing values for each column are zero as shown below.

```
       x              age             sex            city
       0                0               0               0
province          country              v1       confirmed
       0                0               0               0
duration        clean_age          nausea        vomiting
       0                0               0               0
```



K nearest neighbours regression :

Using the clean data for level 2 and 3 with the NAs imputed KNN regression was attempted. Level 1 was left out as it was continually giving higher RMSE. The model was written in *Python* using the scikit-learn library. Using the *labelencoder* all the string categorical variables are encoded to numeric categories, a requirement of *sklearn* regression. The data was split 80/20% into test and validation sets in order to find an optimal K. Iterating over 100 choices of K nearest neighbours we plotted the RMSE to find the optimal K (figure 3.)

Trying various combinations of rows in level 2 we found the RMSE on our predictions ranged on average from 4.7-4.6 with highs of up to 9 RMSE. Our best predictions came with all variables in level 2 at 4.78, and just country, V1, confirmed, cough at 4.58. We similarly got 4.58 using level 3 country, V1,confirmed, respiratory,

Figure 3. RMSE for K = 1:100, Top: Level 3 'country', 'V1', 'confirmed','respiratory' choose 13; Bottom: Level 2 'country', 'V1', 'confirmed','cough' 8 selected

but using all features in level 3 reported up to 13 RMSE.

Support Vector Machine Regression:
Again, *Python* sklearn library was used. *SVM regression* does not auto scale therefore *sklean precposseor's standardscaler* was used to scale all of the data, accounting for differences in scale amongst features. The data was similarly encoded and split into train and validation using RMSE to determine the best kernel for the *svm*. Unfortunately, none of the kernels returned good results, the high number of zero durations pull all the results down to zero or below. Even rounding all negative values up to zero the best score we could obtain was using rbf, *radial bias function*, this is a radial rather than planar kernel and had an RMSE of 6.36.

Note on Score function:
In both svm and knn regression we used the score() function on the test and validation set to look for overfitting and general goodness of fit for the model. This is the coefficient of determination R^2 of the prediction.
Our best knn model gave: Training Score: 0.176, Validation Score: 0.337
Our best svm model gave :Training Score: -2.142, Validation Score: -2.048

Gamma Regression:
Comparing our histograms in figure 2. to the Probability Density Plots in figure 4, it appears that duration follows a *Gamma* distribution. We used the glm() function with the family Gamma() trying both log and identity as our link functions. Using our main concern using this method is that classical models cannot easily handle levels outside of the training data. Almost all columns in the test set contain levels outside of those in our training set. Furthermore, again saw predictions pulling towards zero. Upon further consideration we realized that, although our training

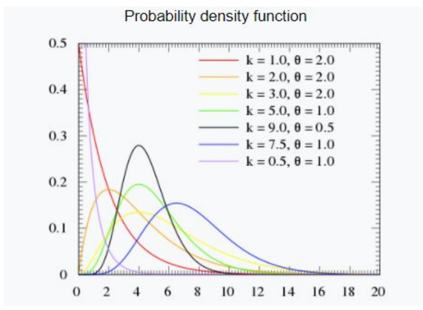


Figure 4.Gamma distribution PDF

duration follows a *Gamma* distribution, perhaps our testing does not

Ridge Regression:
We want to minimize the complexity of the model by using the *Ridge Regression* technique using the glmnet library in *R*. First, the logical value variables were all converted to numeric values of '1' (for TRUE) and '0' (for FALSE). Then, we had to find the best optimal lambda as our parameter(lambda that minimizes the Mean Squared Error), to penalize insignificant predictors in our model (figure 5.)

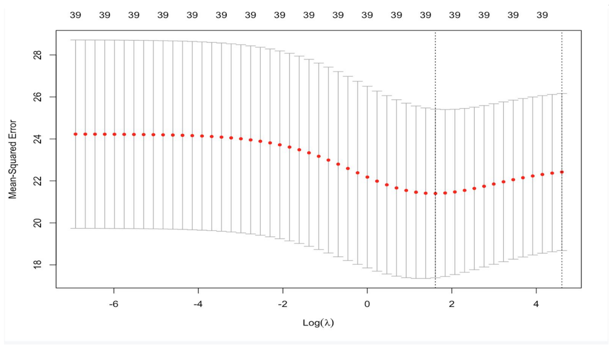
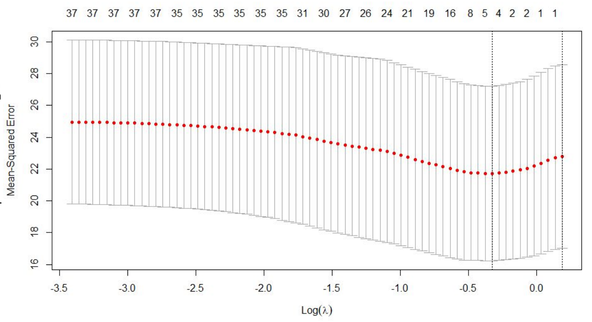


Figure 5. MSE by lambda for Ridge Regression

We first modeled level 2 data with this method, which results in an RMSE of 4.50. On the other hand, using level 3 data results in RMSE of 4.65.

LASSO:
Another method to minimize complexity of the model is using the *LASSO* method from glmnet. Since lambda-1SE produces a too-simple model (only consists of the intercept), lambda-min is used for prediction (figure 6.). Level 1 data results in RMSE of 4.56696 and level 2 data results in RMSE of 4.56656, only a 0.0004 difference. Level 3 data produces a very high RMSE of 47.5.



Random Forest Regression:

Figure 6. MSE by lambda for LASSO Regression

We used the randomForest package in *R*. Plotting the iterated number of trees against the error graph, we want to see which number of trees gives the lowest value of error (figure 7.). Then, We generated the *Variable Importance Plot*, which provides a list of the most significant variables in

descending order by a mean decrease. The top variables contribute more to the model than the bottom ones and also have the highest predictive power (figure 8.)

Based on the variable importance plot, we decided to include confirmed, clean_age, cough, sore.throat, city, province, and fever in our model. We were able to achieve an RMSE of 4.27, which is the lowest value we have so far. However, any models containing an element of randomness may be less reliable as their predictions will vary depending on the seed.

Figure 7. MSE vs number of trees

Neural Networks:

With neural networks, we used the *Python* library fast.ai. Their tabular model takes the categorical variables and passes them through an embedding layer and then a dropout layer, while the continuous variables go through a *batchnorm1d* layer. Once this is done, both layers are combined and fed into some *LinBnDrop* layers and then a *Linear* layer.

For learning rates, we set a maximum learning rate for the cyclical learning rate technique that was developed by Leslie Smith. To determine the maximum learning rate to set, we plotted the losses on the y axis and the learning rate on the x axis, and picked a learning rate in which the slope of the graph was most negative (figure 9.).

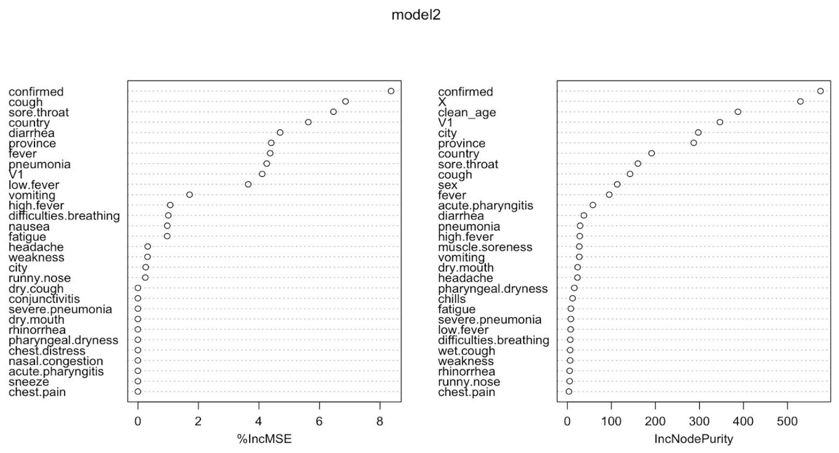


Figure 8. Variable Importance plot for Random Forest Regression

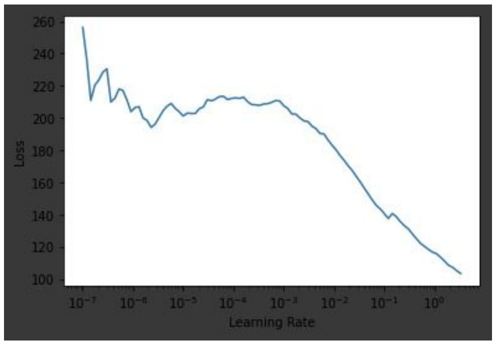


Figure 9. Loss vs. Learning Rate

Using the level 2 data we achieved an RMSE of 4.695

Conclusion:

As seen above, we got similar results with all of the models except the one case in random forest which predicted at a 4.27 RMSE. Variable selection had a noticeable impact on our RMSE scores compared to the model we selected. Our level 2 cleaning, which has symptoms as indicators grouping similar symptoms, or symptoms under aliases, together continually gave us the best prediction.

As a final check we compared the predicted values for our model which achieved 4.27 RMSE with another model using *Random Forest* that did not use sore.throat and instead used V1 and had an RMSE of 4.48. We first looked to see if there was any overfitting, e.g. one always predicting high or low durations, and this did not seem to be the case. We then compared the predicted durations side by side highlighting all differences greater than 3. Looking back at those individuals in the test data there was no clear connection between all of these individuals, except, perhaps, many had no symptoms, although this is true for many individuals in the dataset.

We conclude that there is no overfitting and we have chosen a *Random Forest* Model using confirmed, clean_age, cough, sore.throat, city, province, and fever on our level 2 data resulting in an RMSE of 4.27.