1. Executive Summary

The primary goal of this report is to predict the current speed of COVID-19 spread. As a measure of the speed of spread, the value that we will be predicting would essentially indicate the average number of new infections generated by a single infectious individual at a given time. To predict this rate of spread, we have used the mobility trend data which shows how visits to places like grocery stores, parks, etc. are changing. This dataset includes the mobility trend in a sample locality, across different categories – Grocery & Pharmacy, Parks, Transit Stations, Retail & Recreation, Residential and Workplaces. Data for these 6 categories is captured for up to 11 days, meaning a total of 66 features (6x11), considering the inherent lags in symptom onset of the disease after exposure to the virus.

The average rate of spread as per the given data for 152 days' worth of measurement, is approximately 1.19. It ranges from a minimum value of 0.844 to a maximum value of 2.015. What this means is that on an average a single infectious individual generates at least 1 new infection at a given time, if not more. This rate of spread is affected the most by mobility in the workplace and mobility in places like restaurants, cafes, shopping centers, theme parks, museums, libraries, and movie theaters. The mobility rate in workplaces has been observed to reduce over time. The average mobility in workplaces 11 days back, was around 27.95% less than the baseline value, whereas the same now is around 29.7% less than the baseline. With this gradual decrease in the average mobility, we can hypothesize that some measures must've been taken at workplaces to ensure this reduction. But still, there are some instances when the mobility is as high as 4% more than the baseline in workplaces. It is observed that the rate of spread on days when the mobility in workplaces is more than the baseline (on any of the past 11 days starting from the date of current measurement), is particularly high (approximately 1.8) and the same on days when the mobility in workplaces is lesser than the average value is approximately 1. This shows that if mobility is allowed to increase in just the workplaces even, it would almost double the rate of spread. Mobility in another important category – retail and recreation, that has shown to impact the rate of spread acutely, also shows a very similar trend. The average mobility for this category, 11 days back was around 15.76% less than the baseline value whereas the same now is around 16.61% less than the baseline. Just like in workplaces, even in the retail and recreation segment there are some instances where the mobility rate has seen some unusual highs. Some days record a mobility rate as high as 17% more than the usual baseline in this category. The average rate of COVID spread recorded for days when any of the past 11 measurements of mobility in this segment showed such highs, is found to be around 1.85 and the same on days when mobility on all past 11 days is lower than the average mobility value for this category, is found to be around 1.06, again showing that mobility increase in this segment also approximately doubles the rate of spread. When mobility rate in both, workplace and retail & recreation is above the baseline value, the rate of spread is found to be as high as 1.97. Thus, reducing the mobility in these 2 segments acutely helps in preventing an increase in the rate of COVID spread.

Team Contribution:

Preliminary Data Analysis – Deeksha Aswal, Linear Models – Ritesh Singh,
Tree-Based Approaches – Karan Gupta, Summarization and Evaluation – Shreya Apte

2. Technical Summary 1: Ridge Regression

The objective of this project is to predict the current speed of COVID-19 spread. The dataset consists of 66 features that provide information on mobility changes in places of activity for the past 11 days. As there are multiple features, identifying the most contributing features for estimating Rt is important, for coming up with the model that gives the most appropriate predictions. Keeping this mind, we first fit a linear regression model on all predictors. Based on this model's results, only a few variables were relevant. Hence, to see if any variable's importance is being masked, we checked for multicollinearity between the features using variance inflation factor.

```
> car::vif(linear_fit_all_predictors)
retail_and_recreation_percent_change_from_baseline_11
                                                90.64527
 grocery_and_pharmacy_percent_change_from_baseline_11
                                                17.32399
                parks_percent_change_from_baseline_11
                                                19.50601
     transit_stations_percent_change_from_baseline_11
                                               109.81879
           workplaces_percent_change_from_baseline_11
                                                57.83318
          residential_percent_change_from_baseline_11
                                                85.95023
retail_and_recreation_percent_change_from_baseline_10
                                               140.11679
 grocery_and_pharmacy_percent_change_from_baseline_10
                                                23.25307
                parks_percent_change_from_baseline_10
                                                24.62193
```

```
     transit_stations_percent_change_from_baseline_9
                                              166.62001
           workplaces_percent_change_from_baseline_9
                                               71.20296
          residential_percent_change_from_baseline_9
                                               95.07280
retail_and_recreation_percent_change_from_baseline_8
                                              133.71015
 grocery_and_pharmacy_percent_change_from_baseline_8
                                               24.48023
                parks_percent_change_from_baseline_8
                                               25.82239
     transit_stations_percent_change_from_baseline_8
                                              161.91186
           workplaces_percent_change_from_baseline_8
                                               81.81427
          residential_percent_change_from_baseline_8
                                               98.46175
retail_and_recreation_percent_change_from_baseline_7
                                              111.30356
```

```
 grocery_and_pharmacy_percent_change_from_baseline_6
                                                22.83854
                parks_percent_change_from_baseline_6
                                                23.83218
     transit_stations_percent_change_from_baseline_6
                                               127.67400
           workplaces_percent_change_from_baseline_6
                                                83.05767
          residential_percent_change_from_baseline_6
                                                91.77735
retail_and_recreation_percent_change_from_baseline_5
                                               113.08055
 grocery_and_pharmacy_percent_change_from_baseline_5
                                                22.66786
                parks_percent_change_from_baseline_5
                                                23.44502
     transit_stations_percent_change_from_baseline_5
                                               137.90278
           workplaces_percent_change_from_baseline_5
                                                75.11559
          residential_percent_change_from_baseline_5
                                                83.29707
retail_and_recreation_percent_change_from_baseline_4
                                               135.11175
 grocery_and_pharmacy_percent_change_from_baseline_4
                                                24.35592
                parks_percent_change_from_baseline_4
                                                25.40985
     transit_stations_percent_change_from_baseline_4
                                               172.17217
```

```
           workplaces_percent_change_from_baseline_2
                                                80.34347
          residential_percent_change_from_baseline_2
                                                87.88626
retail_and_recreation_percent_change_from_baseline_1
                                               113.06447
 grocery_and_pharmacy_percent_change_from_baseline_1
                                                15.12049
                parks_percent_change_from_baseline_1
                                                17.47414
     transit_stations_percent_change_from_baseline_1
                                               132.70629
           workplaces_percent_change_from_baseline_1
                                                61.87405
          residential_percent_change_from_baseline_1
                                                72.49196
```

Multicollinearity exists between predictors as indicated by high VIF values. Almost every feature's variance has been inflated due to multicollinearity. Hence to identify the important predictors we tried subset selection and shrinkage methods – Lasso and Ridge. Lasso regression diminishes the insignificant predictors by reducing their coefficients to zero. In Ridge, the least important predictors have coefficients close to zero, but the predictors are still included. Considering that the data contains information about mobility in places of activity for the past 11 days, we assume

that all the features would somehow affect the rate of spread, though the impact of each would vary.
We tried both subset selection and shrinkage models and compared the test mean square error for each model. The least mean square error was obtained using ridge regression model. Lasso model gave a higher mse as compared to Ridge. This supports our initial assumption that ridge model would give more accurate predictions as it includes even the variables which appear to be insignificant, by assigning them a small co-efficient instead of eliminating them.

Below are results for train and test mse using Ridge regression model:

```
[1] "The training MSE as per the Ridge Regression model is:
0.0251444897845398"
[1] "The test MSE as per the Ridge Regression model is:
0.0131347524780487"
```

Using this model, we obtained the following coefficients for the predictors:

predict(ridge_model_train, type='coefficients', s=best_lambda_ridge_train)

```
> predict(ridge_model_train, type='coefficients', s=best_lambda_ridge_train)
67 x 1 sparse Matrix of class "dgCMatrix"
                                                                1
(Intercept)                                          1.802700e+00
retail_and_recreation_percent_change_from_baseline_11  2.970530e-04
grocery_and_pharmacy_percent_change_from_baseline_11  -9.730083e-04
parks_percent_change_from_baseline_11                 -1.224738e-04
transit_stations_percent_change_from_baseline_11       4.637243e-04
workplaces_percent_change_from_baseline_11             9.328041e-04
residential_percent_change_from_baseline_11           -1.408085e-03
retail_and_recreation_percent_change_from_baseline_10  2.942988e-04
grocery_and_pharmacy_percent_change_from_baseline_10  -7.499001e-04
parks_percent_change_from_baseline_10                 -5.835145e-05
transit_stations_percent_change_from_baseline_10       4.470455e-04
workplaces_percent_change_from_baseline_10             9.529788e-04
residential_percent_change_from_baseline_10           -1.694340e-03
retail_and_recreation_percent_change_from_baseline_9   2.311099e-04
grocery_and_pharmacy_percent_change_from_baseline_9   -7.383423e-04
parks_percent_change_from_baseline_9                  -6.624631e-05
transit_stations_percent_change_from_baseline_9        5.932947e-04
workplaces_percent_change_from_baseline_9              9.455615e-04
residential_percent_change_from_baseline_9            -1.449101e-03
retail_and_recreation_percent_change_from_baseline_8   2.189282e-04
grocery_and_pharmacy_percent_change_from_baseline_8   -8.410153e-04
parks_percent_change_from_baseline_8                  -6.704655e-05
transit_stations_percent_change_from_baseline_8        6.081440e-04
workplaces_percent_change_from_baseline_8              1.039179e-03
residential_percent_change_from_baseline_8            -1.803586e-03
retail_and_recreation_percent_change_from_baseline_7   2.117106e-04
grocery_and_pharmacy_percent_change_from_baseline_7   -1.115469e-03
parks_percent_change_from_baseline_7                  -1.109843e-04
transit_stations_percent_change_from_baseline_7        4.766504e-04
workplaces_percent_change_from_baseline_7              7.469233e-04
residential_percent_change_from_baseline_7            -1.309324e-03
retail_and_recreation_percent_change_from_baseline_6   2.988783e-04
grocery_and_pharmacy_percent_change_from_baseline_6   -6.282436e-04
parks_percent_change_from_baseline_6                  -1.072382e-04
transit_stations_percent_change_from_baseline_6        5.541152e-04
workplaces_percent_change_from_baseline_6              7.314861e-04
residential_percent_change_from_baseline_6            -1.249509e-03
retail_and_recreation_percent_change_from_baseline_5   2.315112e-04
grocery_and_pharmacy_percent_change_from_baseline_5   -7.642224e-04
parks_percent_change_from_baseline_5                  -1.276023e-04
transit_stations_percent_change_from_baseline_5        3.635889e-04
workplaces_percent_change_from_baseline_5              6.883565e-04
residential_percent_change_from_baseline_5            -1.231548e-03
retail_and_recreation_percent_change_from_baseline_4   3.667607e-05
grocery_and_pharmacy_percent_change_from_baseline_4   -6.828658e-04
parks_percent_change_from_baseline_4                  -2.503042e-04
transit_stations_percent_change_from_baseline_4        9.540862e-05
workplaces_percent_change_from_baseline_4              5.902839e-04
residential_percent_change_from_baseline_4            -8.787144e-04
retail_and_recreation_percent_change_from_baseline_3  -1.203475e-04
grocery_and_pharmacy_percent_change_from_baseline_3   -8.453751e-04
parks_percent_change_from_baseline_3                  -2.559828e-04
transit_stations_percent_change_from_baseline_3        4.142645e-05
workplaces_percent_change_from_baseline_3              6.541547e-04
residential_percent_change_from_baseline_3            -9.371063e-04
retail_and_recreation_percent_change_from_baseline_2  -4.203569e-04
grocery_and_pharmacy_percent_change_from_baseline_2   -8.717610e-04
parks_percent_change_from_baseline_2                  -2.304724e-04
transit_stations_percent_change_from_baseline_1       -1.070149e-04
workplaces_percent_change_from_baseline_1              9.297161e-04
residential_percent_change_from_baseline_1            -6.143840e-04
>
```

2. Technical Summary 2: Extreme Gradient Boosting - XGBoost

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. As we know, in boosting the idea is to sequentially build models by minimizing the errors from previous models while increasing (or boosting) influence of high performing models.
XGBoost expands on a specific class of boosting methodology which is the Gradient Boosting method. Gradient Boosting is a special case of boosting, in which errors are minimized by utilizing the Gradient Descent algorithm wherein, at each step a tree is created to predict the residuals, and then the new residuals are added to the previous trees' predictions to update the prediction towards a more accurate prediction. XGBoost expands on Gradient Boosting by combining exceptional software and hardware optimization techniques to yield superior results using fewer computing resources in the shortest amount of time. Below are a few improvements that XGBoost offers over general boosting methodologies -

1. Parallelized Tree Building
2. Tree Pruning using Depth first approach - The stopping criterion for tree splitting within GBM framework is greedy in nature and depends on the negative loss criterion at the point of split. XGBoost uses 'max_depth' parameter as specified instead of criterion first and starts pruning trees backward. This 'depth-first' approach improves computational performance significantly.
3. Regularization – XGBoost penalizes more complex models through both Lasso (L1) and Ridge (L2) regularization to prevent overfitting.

XGBoost uses a large set of hyperparameters, which we have computed using a grid of value ranges for each parameter, and then utilized the value set that gives the best prediction as the final hyperparameters for our final model. The hyperparameter grid is as follows:

```
xgbGrid <- expand.grid(nrounds = c(100,200), #maximum number of trees
                       max_depth = c(5, 10, 15, 20, 25), #maximum tree depth
                       colsample_bytree = seq(0.2, 0.9, length.out = 5), #column sampling
                       eta = seq(0.1, 0.9), #learning rate for adjusting weights at each step
                       gamma=c(0, 0.05, 0.1, 0.5),
                       min_child_weight = c(1,2,3),
                       subsample = c(0.5, 0.75, 1.0) #row sampling
)
```

The hyperparameters reported by our XGBoost model, which resulted in the best error estimation when fitted on the train data, were:

```
xgb_model$bestTune
```

```
  nrounds max_depth eta gamma colsample_bytree min_child_weight subsample
2     200         5 0.1     0              0.2                1         1
```

Here:
nrounds indicates number of iterations,
max-depth indicates maximum tree depth for each iteration,
eta is the learning rate,
gamma controls regularization,
colsample_bytree controls number of features supplied to a tree,
min_child_weight refers to the number of minimum instances required in a child node,
subsample controls number of samples supplied to a tree.

Using XGBoost, we obtained below train and test errors after fitting the model for the best set of hyperparameters:

```
set.seed(122)

xgb_model = train(
  X_train, y_train,
  trControl = xgb_trcontrol,
  tuneGrid = xgbGrid,
  method = "xgbTree",
  objective ='reg:squarederror'
)

xgb_model$bestTune

predicted_train = predict(xgb_model, X_train)
train_mse_xgb = mean((predicted_train - y_train)^2)

predicted_test = predict(xgb_model, X_test)
test_mse_xgb = mean((predicted_test - y_test)^2)

print(paste("The Training MSE for Xtreme Gradient Boosting is:",round(train_mse_xgb,5)))
print(paste("The Test MSE for Xtreme Gradient Boosting is:",test_mse_xgb))
```

[1] "The Training MSE for Xtreme Gradient Boosting is: 0"
[1] "The Test MSE for Xtreme Gradient Boosting is: 0.005966368"

3. Technical Summary 3: Random Forest

Since, our main goal is prediction, we don't much care about the interpretation of the model. This gives us leeway to try more flexible models. Hence, we first tried Decision trees. Decision trees although were great for interpretability, we faced the issues that Decision trees had high variance. Hence to deal with high variance, we decided to move on to Random Forest, which is a special case of Bagging.

In Random forest, there are two parameters which we need to tune in order to get the least error. One is ntree and other is mtry (which indicates the number of variables used in each tree). In general, as we increase the number of trees, the variance in the model decreases, hence we choose a model with ntree = 1000.

Further, to find the most optimum value of mtry, we try the models for various values of mtry using the training data set and calculate the respective Test MSE. We select the value of mtry corresponding to the least value of test MSE.

```
> for(mtry in 1:ncol(X)){
+   set.seed(122)
+   rf_fit = randomForest(Mean.R.~.,
data=train, mtry=mtry, ntree=1000)
+   prediction_rf = predict(rf_fit, test_X)
+   test_error_rf[mtry] = mean((prediction_rf - test_Y)^2)
+ }
> mse_rf = min(test_error_rf)
> mtry_for_min_mse = which.min(test_error_rf)
> mtry_for_min_mse
[1] 3
> print(paste("The Mean Squared Error for Random Forest is:",mse_rf,",
observed for mtry:",mtry_for_min_mse))
[1] "The Mean Squared Error for Random Forest is: 0.0056638422470266
, observed for mtry: 3"
```

We get it for mtry = 3. We can see the corresponding values of training error and test MSE below-

```
> # Building the model with mtry = 3.
> set.seed(122)
> rf_fit = randomForest(Mean.R.~., data=train, mtry=3, ntree=1000)
> prediction_rf = predict(rf_fit, train_X)
> training_error <- mean((prediction_rf - train_Y)^2)
> training_error
[1] 0.00255836
> prediction_rf = predict(rf_fit, test_X)
> test_MSE = mean((prediction_rf - test_Y)^2)
> test_MSE
[1] 0.005663842
```

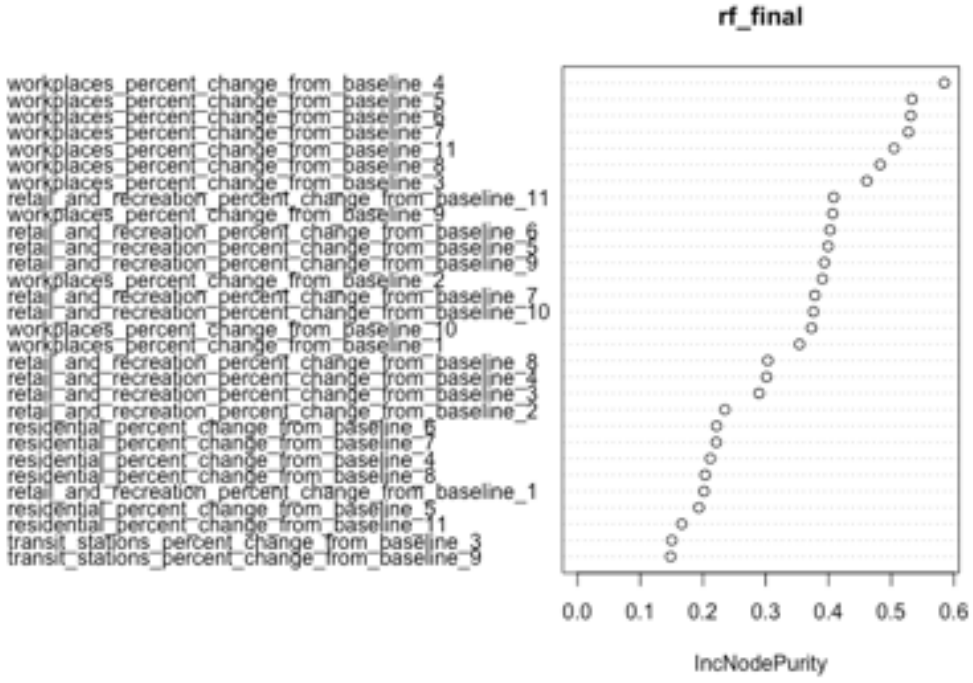
We see that the training error is less than the test error, which is understandable since model aims to decrease the training error. We now explore the final model using complete data.

```
> rf_final

Call:
 randomForest(formula = Mean.R. ~ ., data = data, mtry = 3, ntree =
1000)
               Type of random forest: regression
                     Number of trees: 1000
No. of variables tried at each split: 3
          Mean of squared residuals: 0.01039401
                    % Var explained: 88.18
```

So here we can see that the tree tries three variables for each split. This helps in creating de-corelated trees. We also see that the forest is able to explain approximately 90% of the variation in the data.
It is also important to find out which variables have greater impact on the model. We achieved the same using the variable importance. We see the plot for the same below.

```
> varImpPlot(rf_fit)
```



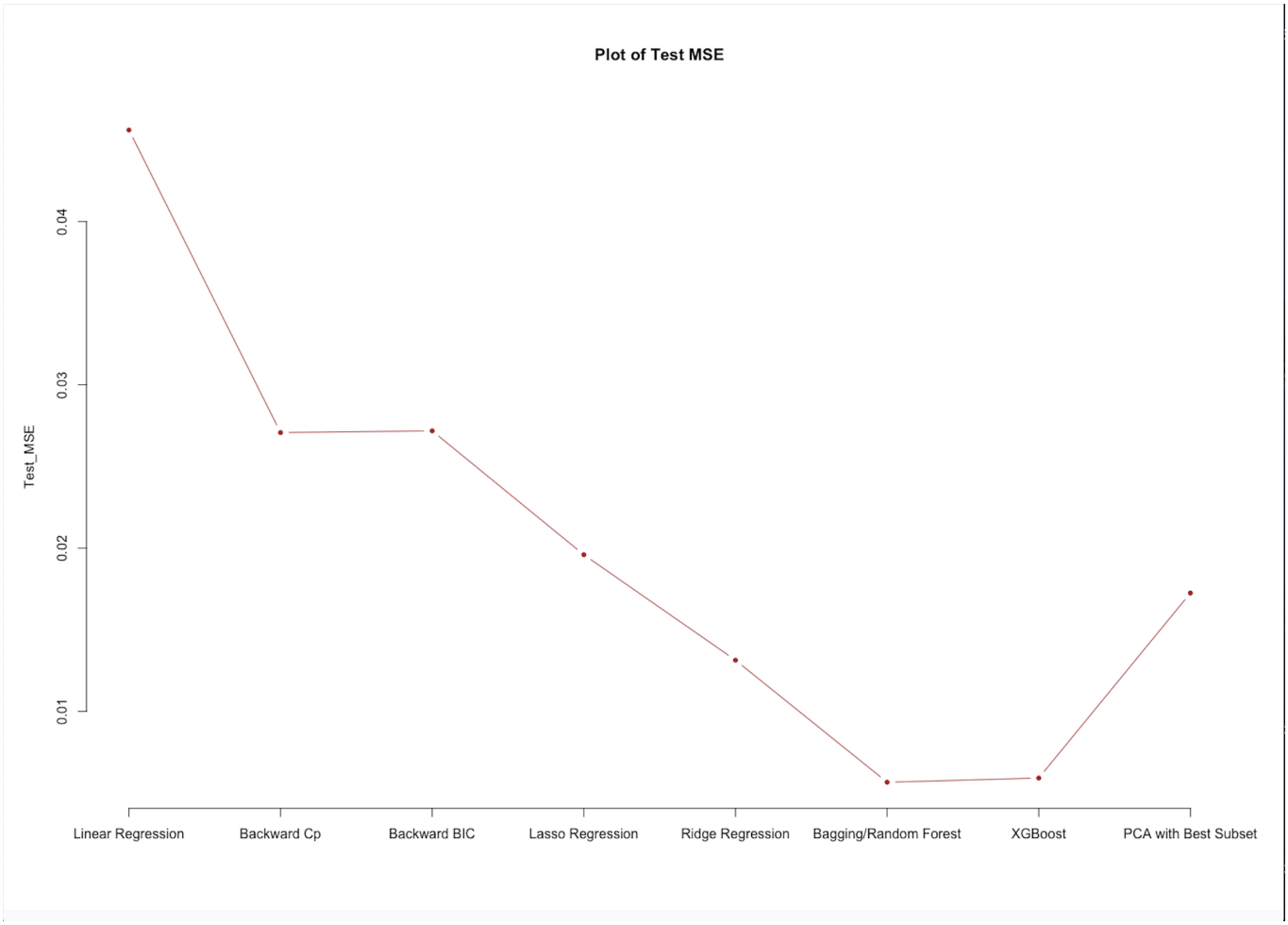
The plot above shows the variables according to their relative importance in the random forest. Here we see that percentage change in workplaces are leading factors which affect the rate of spread. This is followed by retail and recreation and lastly by residential percent change from base line.
So, for final model will be trained using complete data and value of mtry = 3.

4. Choosing the best model

The final aim of the project is to most accurately predict the rate of spread basis the given features, for which we have to minimize the mean squared error of the predictions on the test data from our chosen model.

So, to evaluate the best model, we have chosen the validation set approach in which we have split the given training data into 70% train set and 30% validation set. Using the 70% train set as the training data while fitting the models, we have estimated the test/validation mse on the remaining 30% validation set.

In doing so, we discovered that the Random Forest model generates the lowest test/validation mse as can be seen in the graph below. Also, with the output of the Random Forest model we can identify the most important features that were utilized in reaching an accurate prediction.



5. Evaluation of Best Model using Test Dataset

```
test_data = read_excel('R_calculated_v2_11_Day_Interval_test.xlsx')
test_data_X = test_data[,!names(test_data)=='Mean(R)']
test_data_Y = test_data$`Mean(R)`

prediction_rf_final = predict(rf_final,test_data_X)
test_mse_rf = mean((prediction_rf_final - test_data_Y)^2)
print(paste("The Test Mean Squared error using our best model is:",test_mse_rf))
```

"The Test Mean Squared error using our best model is: 0.00978942257643382"

6. Introducing a better model

After a deeper analysis of the hyperparameters, we identified a margin of improvement in our Extreme Gradient Boosting Model. By increasing the number of computations for selecting the hyperparameters (which we achieved by adding a larger margin of values in the hyperparameter grid), our XGBoost model provided an even smaller train, validation as well as test error for the provided train and test datasets.

The hyperparameters that we allowed tuning modification for include:

1. nrounds – which indicates the number of allowable iterations. We provided an array of higher values for this to account for a greater number of iterations. This improved our score.

The modified hyperparameter grid that led to increase the accuracy of out XGBoost model is as follows:

```
set.seed(122)
xgbGrid <- expand.grid(nrounds = c(400,600,800), #no of iterations
                       max_depth = c(5, 10, 15, 20, 25), #maximum tree depth
                       colsample_bytree = seq(0.2, 0.9, length.out = 5), #column sampling
                       eta = seq(0.1, 0.9), #learning rate for adjusting weights at each step
                       gamma=c(0, 0.05, 0.1, 0.5), #regularization parameter
                       min_child_weight = c(1,2,3), #minimum number of instances needed to be in each node
                       subsample = c(0.5, 0.75, 1.0) #row sampling
)
```

Using this grid of values, we fit one XGBoost Model for each parameter value set on the entire train data, and then predicted the rate of spread using each model for the provided test data. We stored the test mse for each such model in an array and selected the minimum value of test mse from this array to be our final test mse from XGBoost.

After that, to fixate one model, we selected the parameter value set which led to this increased accuracy and fit the final XGBoost Model using this fix hyperparameter value set for further predictions. This final model is our best-chosen model for this Problem Statement.

The test mean squared error achieved by our final, parametrized XGBoost Model is as follows:

```
set.seed(122)
predicted_test_data = predict(xgb_model_best_grid, data.matrix(test_data[,-1]))
test_mse_xgb = mean((predicted_test_data - test_data$`Mean(R)`)^2)
print(paste("The Final Test MSE for Xtreme Gradient Boosting is:",test_mse_xgb))
```

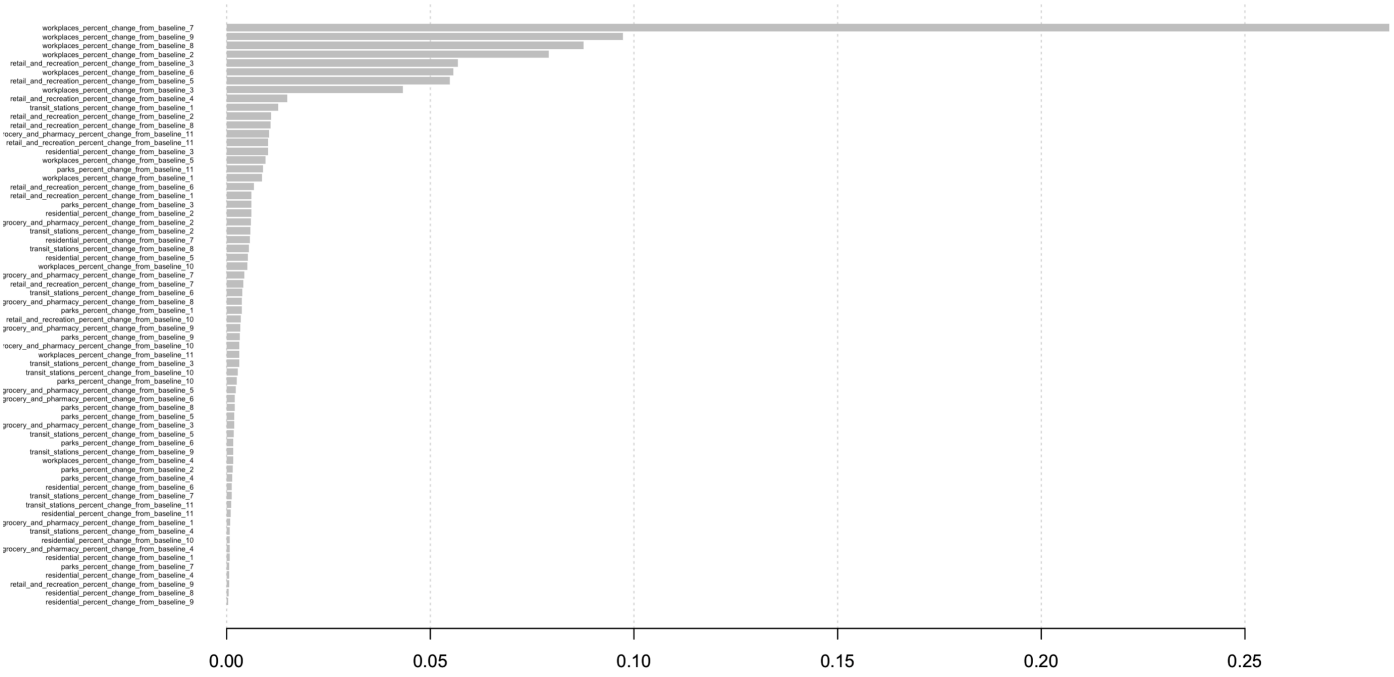
"The Final Test MSE for Xtreme Gradient Boosting is: 0.00373547855014643"

Thus, the final test MSE that we would like to report for our project is - **0.00373547855014643**
This test mse can be reproduced by running the code in the attachment – Best_Model.R in the submitted package directly.

The XGBoost package also provides us with a feature to find the most important predictors, that have contributed to improving the accuracy of the branches that have resulted by splitting on those predictors. Our final model showed the below variables as important ones for achieving this accuracy.

```
### 3.5 Important Features that contribute in improving accuracy brought about by them
#to the branches they are on.

importance_matrix = xgb.importance(colnames(data[,-1]), model=xgb_model_best_grid)
xgb.plot.importance(importance_matrix = importance_matrix)
```



This plot shows that percentage change in mobility in workplaces for up to t-9 days, percentage change in mobility in retail and recreational areas like restaurants, cafes, shopping centers, theme parks, museums, libraries, and movie theaters for up to t-5 days have the most impact on accurate decision making for the rate of spread. This shows that the mobility rate in these categories has been the biggest contributor in the change of the speed of spread of COVID-19. Lesser mobility in these regions prevents the speed of spread from increasing whereas increased mobility raises the speed of spread.