Forecasting Coronavirus:

Developing a time-invariant model to predict outbreaks or subsidence of COVID-19 in the United States

EE 660 Course Project (Type 1)

Arjun Viswanathan

arjunvis@usc.edu

12.05.2020

1. Abstract

Since the first report on its transmission to humans in Wuhan, China, in December 2019, the coronavirus has spread extremely rapidly throughout the world, affecting the lives of millions. As the virus spread, numerous studies have been conducted and published attempting to isolate indicators and causes as to how and why the disease spreads so quickly. A large subset of early studies postulated that climate, along with more obvious factors such as population density and socioeconomic statistics of that population, was a driving force in spread. This, at least in the United States, led to the notion that the summer heat would kill off the virus [1]. Of course, we now know that did not happen, and the coronavirus continued to spread at high rates throughout the summer and into the fall. However, there are correlations between weather conditions and the virus spread, especially with temperature, though the extent and even sign of correlation is undetermined. I approach the problem by attempting to fit weather patterns to the percentage growth rate in cases, as opposed to absolute values or per capita growth rates, as to account for both population discrepancies and even the time of the sample, as the percentage growth rates generally remain around the same value. Using both regularized regression and tree-based regression models, I analyze daily weather patterns on a county level to predict county level growth rates. The results of this analysis proved to be generally inconclusive, but there is promise in model improvement over a simple linear fit.

2. Introduction

2.1. Problem Type, Statement and Goals

The aim of this project is to determine, with a much larger sample size of data, what factor weather can have in the prediction of virus spread. The other studies I have found on the subject have tried to fit linear regression models of local weather data to the cumulative value of cases in that local region. Rather than look at cumulative cases, I am attempting to find a relationship that is more invariant to time and sociological factors, by fitting the model against the *daily percent increase* in cases. Along with reducing the impact of population on the output, this method will also better linearize the output features, since the spread of disease is normally exponential in growth. This is a regression problem, whereby the input variables of temperature, precipitation, and weather conditions will be used to predict the percent increase of cases for that data point. Each data point corresponds to a US county on a specific day.

This data requires a large amount of preprocessing, with many missing values for cases and weather data that need to be imputed. Along with that, I am adding additional features to the original dataset, such as climate region zones and changes in numeric weather statistics (% day-by-day change in temperature and precipitation).

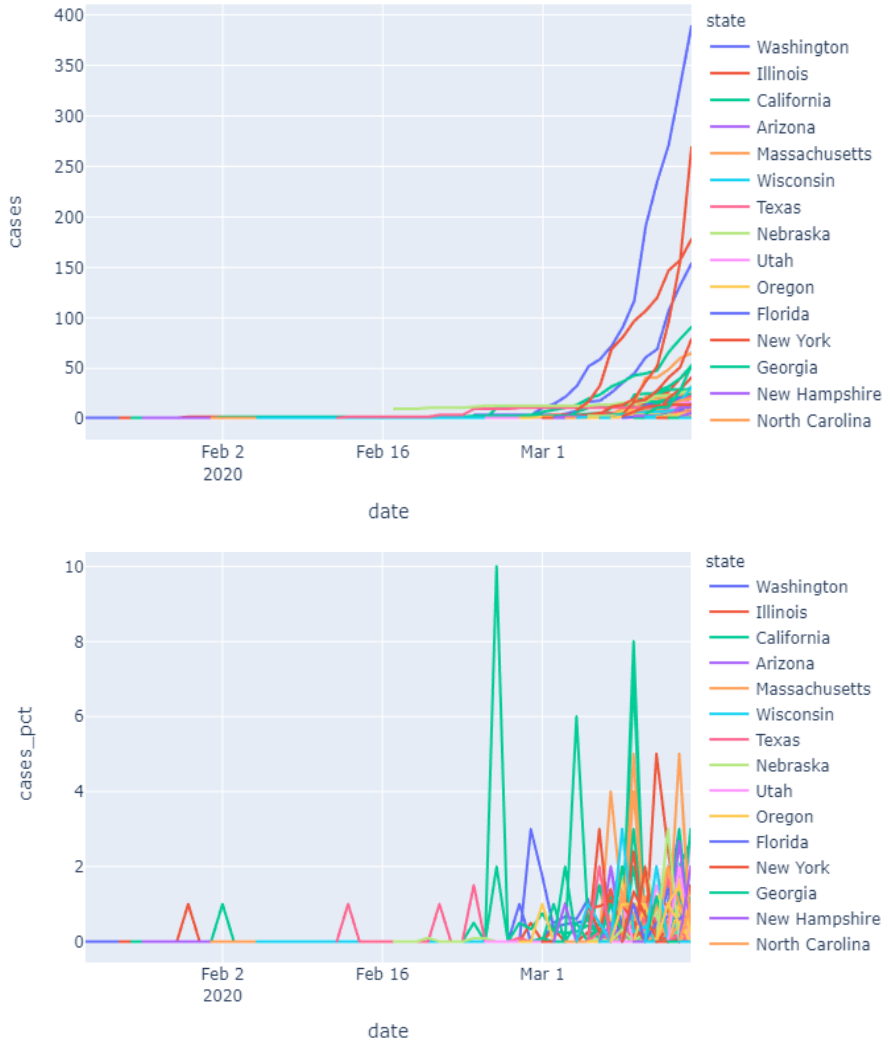

Figure 1: Cumulative cases by county from January 1, 2020 to March 14, 2020 (top), day-by-day percent increase in cases over same period

2.2 Literature Review

Since March 2020, there have been numerous studies on the correlation between different weather patterns and the spread of coronavirus, and I found a couple to be particularly interesting. In an aggregate study published in August, the variables humidity, precipitation, radiation, temperature, and wind speed were all found to have conflicting conclusions on their correlation with COVID-19 spread across the 61 studies aggregated [2]. Temperature was particularly noteworthy, as roughly half of the studies found a negative correlation, while several others found a positive correlation, with roughly a third finding no association or an

inconclusive result. Another wide-range study done in Mexico found that humidity played a crucial, and statistically significant, role in spread, but is inextricably linked with temperature and precipitation among the regions studied [3]. As I do not have access to humidity data, hopefully temperature and precipitation may act as a proxy for that variable. This study also took into consideration the transmission phase, and only used data from Mexico's "Phase 1", when there were few to no policies in place to curb spread.

2.3 Prior or Related Work - None

2.4 Overview of My Approach

My attempt to correlate predict weather hinges on transforming cumulative case counts into daily percent increases. As shown in Figure 1, this transforms an exponential growth into a more uniformly varied set of values. By measure change instead of case counts, I also account for time and population to an extent, as day-by-day growth rates for an exponential function like disease spread become values roughly around 1, regardless of the current number affected until a limit is reached (at which point growth slows). Through this, the weather of a local region can be safely compared across the whole input space without considering the time-varied factors of the output like the population affected.

I also postulate that smoothing the weather over a period of time may result in better predictions, under the assumption that cases are not reported when a person is immediately affected by coronavirus, and that there is a window period of a few days where a person could get the coronavirus before being diagnosed. As such, the weather over that few days period would affect the growth rate more than the weather on that day alone. In my approach, I add a hyperparameter *rolling_period* to account for that, which is described in better detail later.

3. Implementation

3.1 Data Set

Originally, data gathering method was direct: I downloaded COVID-19 county case data from the New York Times repository [4], and manually inputted US climate region codes as a dictionary, with the information coming from noaa.gov [5]. The weather data proved extremely difficult to download. There is no specific dataset that gives daily weather statistics by US county. Rather, there is sparse data, found in fixed width files, for each US weather station, and county weather data must be extrapolated from the stations located within the county. I had a lot of trouble implementing this part, until I very fortunately came across a Kaggle dataset containing the exact information I needed [6]. Since this dataset was already formatted as a csv, with all data on a county-level scope, it was much easier to import and use only the columns needed, which are as follows:

Feature	Type	Description
date	str -> datetime	Day of observation
county	str	Name of US county
state	str	Name of US state, used to join with climate regions data
fips	str	County FIPS code
cases	int	Cumulative number of reported coronavirus cases in that county on that date
mean_temp	float	Mean temperature on that day in that county (Fahrenheit), affected by *rolling period*
min_temp	float	Minimum recorded temperature on that day in that county (Fahrenheit), affected by *rolling period*
max_temp	float	Maximum recorded temperature on that day in that county (Fahrenheit), affected by *rolling period*
dewpoint	float	Recorded dewpoint temperature on that day in that county (Fahrenheit), affected by *rolling period*
precipitation	float	Amount of precipitation on that day in that county (inches), affected by *rolling period*
fog	int -> bool	Boolean indicated if fog was present on that day
rain	int -> bool	Boolean indicated if rain was present on that day
snow	int -> bool	Boolean indicated if snow was present on that day
hail	int -> bool	Boolean indicated if hail was present on that day
thunder	int -> bool	Boolean indicated if a thunderstorm was present on that day
tornado	int -> bool	Boolean indicated if tornado was present on that day
region	str	Nominal categorical feature representing one of 9 US climate regions

Table 1: Features extracted from dataset

Additionally, my feature engineering produces the following parameters:

Feature	Type	Description
mean_temp_pct	float	Day-by-day % change of mean temperature on that day in that county, affected by *rolling period*
min_temp_pct	float	Day-by-day % change of minimum recorded temperature on that day in that county, affected by *rolling period*
max_temp_pct	float	Day-by-day % change of maximum recorded temperature on that day in that county, affected by *rolling period*
dewpoint_pct	float	Day-by-day % change of recorded dewpoint temperature on that day in that county (Fahrenheit), affected by *rolling period*
precipitation_pct	float	Day-by-day % change of amount of precipitation on that day in that county, affected by *rolling period*
cases_pct	float	Day-by-day % change of recorded cases in a certain county, *unaffected* by *rolling period* and **used as output variable**

Table 2: Features engineered from existing features

3.2 Dataset Methodology

I split the data into 4 sections: Pretraining, Training, Validation, and Testing data. My original project scope was cases between March 15 and October 31. This was based off the fact that case rates before COVID-19 was declared a global pandemic (March 11) were much lower and more sparsely populated. Thus, I used all data before March 15 as my pretraining data, and the rest of the data up to October 31 was used for training, validation, and testing. The pretraining data is mainly used for exploratory data analysis, as well as finding an optimal PCA value. I chose not to use data after October 31 as, at the time of writing, there are possible discrepancies in the newly reported data (usually too few reported cases).

My goal for this project is to find a time-invariant correlation between weather and cases rates. As such, I split the testing and validation sets from the training set randomly. First, I randomly split off 10% of all data points in the set between March 15 and October 31. Then, of the remaining data, I split another 10% off for the validation set. I also use 3-fold cross validation for hyperparameter optimization.

This system has 2 stages of validation: cross validation and a validation set. Cross validation is used for hyperparameter tuning; these hyperparameters are model-dependent, such as *lambda* for regularized linear regression or *max tree depth* for tree models. These models are scored based on R2 score, to best explain the highly variant data. The validation sets are used for dataset parameter tuning. This parameter is the rolling period, which corresponds to the number of days over which to average numerical input features. For example, a rolling period of 6 would result in the mean_temp, min_temp, max_temp, dewpoint, and precipitation values for each sample being averaged over a 6-day period prior to that sample's date for that county. For each model and each rolling period (from 1 to 7 days), cross validation is first run to determine the best model hyperparameters, and then the model with the best R2 score (across the 7 rolling periods) is selected. Each of those models (1 for each model, 6 in total) are tested against the validation set to determine the best model out of the 6 (baseline included). The validation test is scored with *mean squared error*, as the overall goal is to best predict day-by-day case growth and will give an out of sample error bound with a Hypothesis Set = 6 and N = $N_{Validation}$. The final model is used on the training set, again being scored with *mean squared error*. The best baseline estimator will also be tested on the testing set but, unless it is the best estimate from validation, will not be considered part of the Hypothesis Set. This will give an out of sample error bound with a Hypothesis Set = 1 and N = $N_{Testing}$.

The number of data points in each set (after preprocessing) are as follows:

$N_{Pretraining}$	2088
$N_{Training}$	489180
$N_{Validation}$	54354
$N_{Testing}$	60393

Table 3: Number of samples in each cleaned dataset

3.3 Preprocessing, Feature Extraction, Dimensionality Adjustment

The dataset that my data is drawn from already lightly cleans some data for the purposes of joining multiple data sources together, but there is still some preprocessing required. Firstly, there are several thousand samples where all numerical values (temperatures, dewpoint, precipitation), are missing, accounting for ~6% of all data. These samples can be safely removed without affecting the size of the datasets drastically. The next step of preprocessing is extrapolating missing temperature values. For samples where 2 of the 3 temperature (min, max, mean) values exist, the third can be generally extrapolated. I assume that the mean temperature is roughly equal to the median temperature and is halfway between the minimum and maximum temperatures of that day. Next, precipitation alone has significant amounts of missing data. In this case, I assume that missing precipitation data is equivalent to no precipitation on that day and set all missing values to 0. Dewpoint requires a slightly more complicated method of filling data. Since it is a temperature value, setting missing values at 0 would not make sense, as 0 degrees is more than likely nowhere close to the true value. Also, the overall values have an extremely large variance (mean: 37, variance: 182 on pretraining data), so replacing each NaN value with the overall average would also be a poor estimate. Instead, I replace each missing value with the average dewpoint for that specific county and discard the samples for which there are no values after the transformation (~300 samples). For Boolean values, such as snow, thunder, etc., I assume that a missing value is the equivalent to the absence of that event occurring and, in the same vein of the precipitation adjustment, set the value to 0, or False.

After cleaning the data, additional features are added. These features are cases_pct, mean_temp_pct, min_temp_pct, max_temp_pct, dewpoint_pct, precipitation_pct. These features are extracted as the day-by-day % change in the corresponding variable for that county. For example, in Los Angeles County, a mean_temp of 40 on March 20 and a mean_temp of 50 on March 21 would corresponding to a value of 0.25 for mean_temp_pct in Los Angeles County on March 21. For instances where there is no previous value (leading to NaN), the value is imputed with 0. For instances where a value goes from 0 to 1, or 1 to 0, leading to values of inf and -inf, the values are replaced with 1 and -1, corresponding to a 100% increase or decrease.

For the data to correctly function in PCA and training, the categorical values are transformed to one-hot encoded columns (9 additional columns), and the numeric columns are scaled via StandardScaler().

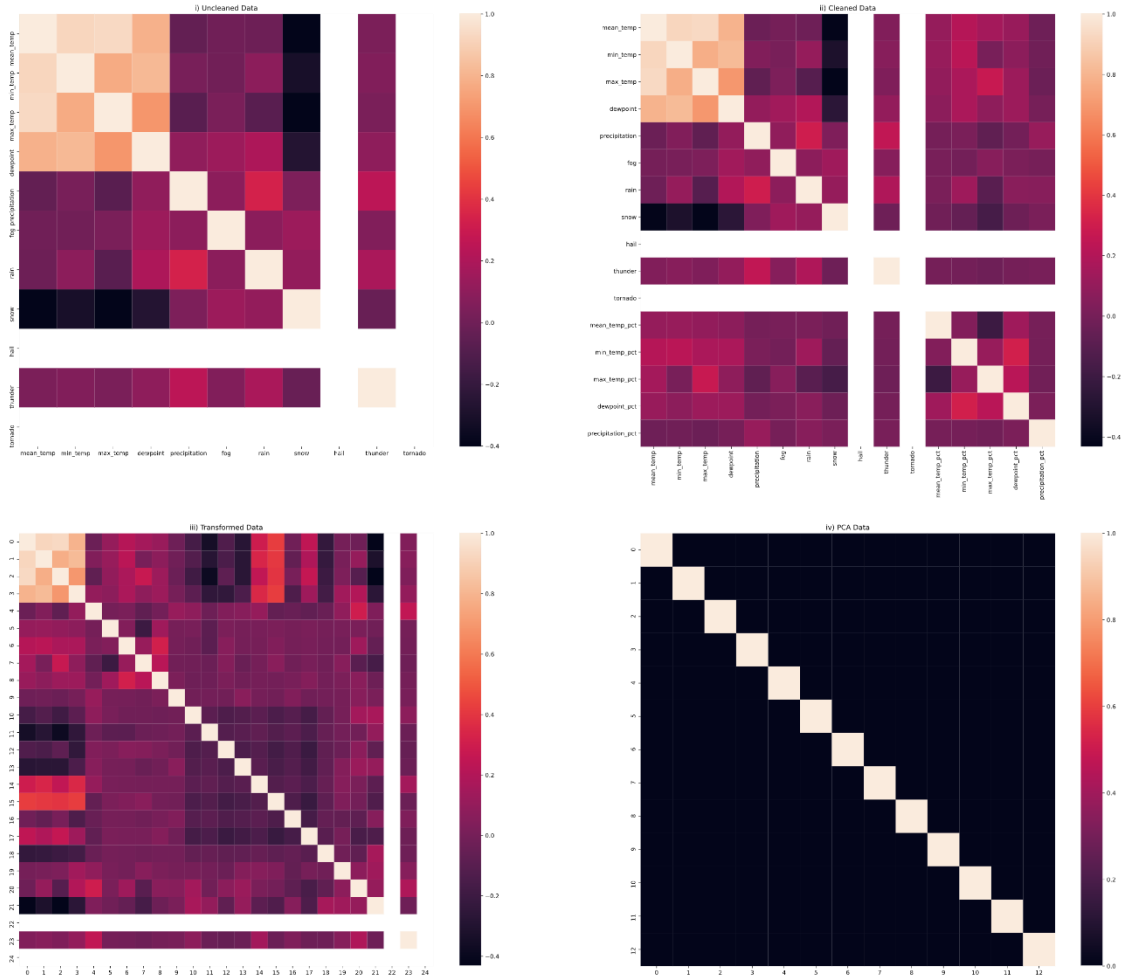

Figure 2: Correlation Matrix of Features before cleaning (top left), after cleaning (top right), after transformation (bottom left), and after PCA (bottom right). Note that the PCA Scale begins at 0, while the rest begin at -0.4

I perform Principal Component Analysis on the data as well, though it is also used as one of the hyperparameters in cross validation for some models. This was done mainly to address the multicollinearity in the temperature values, as seen in Figure 2. First, I found the optimum number of components that would explain at least 95% of the variance in the pretraining data (red line in Figure 3) and used that PCA value (pca_opt = 13) to reduce the features in my tree ensemble and regression regularization methods. In those methods, one of the hyperparameters tested is applying PCA (with pca_opt) or not applying PCA at all. In addition to those methods, I have a baseline linear regression method with no regularization or PCA, and a linear regression method with different levels of PCA.

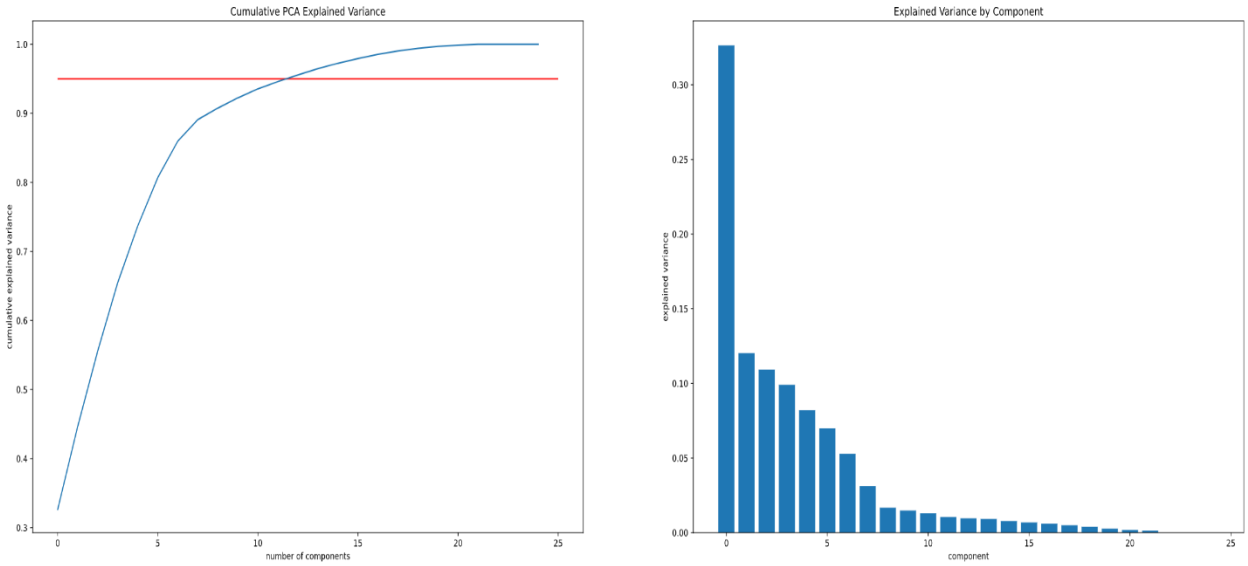

Figure 3: Cumulative variance explained by PCA components (left), explained variance by component (right)

3.4 Training Process

3.4.1 Baseline Model

This model is a simple linear regression model with no hyperparameters or PCA applied and is used as a baseline estimate to compare against the other models.

3.4.2 PCA Model

This model is also generally more of a baseline model. I apply PCA with a varying number of components, ranging from 2 to 17 principal components. Then I apply the transformed data to a simple linear regression model.

3.4.3 Ridge Regression Model

This model is linear regression with a ridge (L2) penalty term. I use this model, in conjunction with PCA, to reduce the multicollinearity of some of the variables, specifically the temperature inputs, which are all very highly correlated to each other. I opted for a L2-regularized penalty term, since there are far fewer features than data samples. As such, the model should not overfit the data, which would require a L1 penalty to reduce the feature size.

For hyperparameter tuning, the only parameter to tune is the alpha (or lambda) value, corresponding to the strength of the regularization term. I used Randomized Search Cross Validation to search across alpha values between 0.01 and 10, on a log uniform scale. Larger values of alpha correspond to stronger penalty terms, and a more conservative model with smaller weights.

3.4.4 Elastic Net Regression Model

This model is an elastic net model, combining both L1 and L2 regularization. This model is trained both with and without transforming the data with PCA, with PCA (or no PCA) acting as a hyperparameter along the search grid. Since elastic net regularization includes an L1 penalty term, it is possible to reduce the feature space, as well as the multicollinearity between

temperature values, without first applying PCA, so I opted to search across a parameter space that tests the effect of PCA or lack thereof.

Like in the Ridge Regression Model, Randomized Search Cross Validation is used to determine the optimal hyperparameters. In this case, there are 3 hyperparameters of interest: PCA, alpha and l1_ratio. The alpha value is functionally the same as before, as it determines the strength of the regularization penalty terms. The PCA hyperparameter determines if PCA (using 13 components as the optimum found in the pretraining set) is used before fitting the model. The L1 ratio determines the ratio of the strength of the L1 regularization penalty to the L2 regularization penalty. With an l1_ratio = 1, elastic net acts like an entirely L1 regularized model, while l1_ratio = 0 causes the model to behave like a L2 regularized model.

I used the same log uniform range between 0.01 and 10 for alpha, while I set l1_ratio to be a uniform distribution between 0 and 1, so I could search the whole space of regularization penalties.

3.4.5 Gradient Boosted Decision Tree (w/ CART) Model

This model is a gradient boosted decision tree, using the XGBoost tree model architecture. While gradient boosting normally minimizes an exponential loss function to determine the optimal weight and regressor at each iteration, XGBoost approximates the loss function via its second order derivative, which leads to faster processing. Additionally, XGBoost ran at a faster rate than the Sklearn GradientBoostRegressor over the same parameters on the pretraining data when both were parallelized. Other than the modified loss function, though, XGBoost operates the same as a Forward Stagewise Additive Model that stacks many weak linear models into a tree.

The hyperparameters of interest in the gradient boosting model are learning_rate, min_child_weight, gamma, subsample, colsample_bytree, and max_depth, along with the same hyperparameter of PCA (13) or no PCA. The learning rate corresponds to the weight of each weak learner iteration on the overall tree, and smaller values lead to a smaller iterative change for each tree in the model. For performance, I opted to maintain the number of trees (or number of iterations) at 100, so lower learning rates may result in a worse loss score. The minimum child weight defines the minimum sum of weights of all the observations within a child node, where higher values would reduce overfitting specific instances found in the training set. Essentially, it corresponds to the minimum size of samples within a node to determine whether that node can be split. I chose a uniform integer distribution between 1 and 10 to determine if there are any specific cases of overfitting within the model, though due to the ratio of data to features, I expect the values to be closer to, if not exactly, 1 for the optimal model. The gamma value is a stopping parameter, and the model only splits a node when it leads to a reduction in the loss function by at least gamma. A gamma value of 0 would cause the tree to split nodes to purity or until another stopping parameter is triggered. High gamma values would result in more conservative models but could lead to underfitting instead of overfitting. I chose a truncated normal distribution ranging from 0 to 5, centering around 0.5, for my gamma value. Subsample and column sample by tree are performance parameters. Subsample denotes the fraction of samples that are randomly sampled from the dataset for each tree, while column sample by tree denotes the fraction of parameters that are sampled for each tree. These values

should typically range from at least [0.5, 1], though I opted for a tighter bound on column sample by tree due to the small number of features in comparison to the number of samples. Subsample is a uniform distribution between 0.5 and 1, while column sample by tree is a uniform distribution between 0.6 and 1. Finally, max depth is the maximum depth of tree, where smaller values would control overfitting. I opted for a uniform integer distribution between 3 nodes and 8 nodes of depth. 2 nodes seems to be way too small of a depth for a tree, while nodes beyond a depth of 8 may not provide much useful information for a feature set of 25 values (after one-hot encoding).

3.4.6 Random Forest Model

This model is a random forest model, again using the XGBoost architecture, modified for random forest trees. Since random forest models are an aggregate (bagging) of separate trees, instead of a single tree iteratively learning from weaker trees, the learning rate is set to 1, with each weak learner having the same relative weight. Again, for performance, I chose to maintain the number of trees at 100, so the final model prediction of a particular input will the average prediction over 100 trees.

Aside from a fixed learning rate, the hyperparameters for the random forest model are roughly the same as the hyperparameters for the gradient boosting model: gamma, max_depth, min_child_weight, subsample, colsample_bynode, and PCA. I kept the values for gamma (truncated normal distribution from 0 to 5, centered at 0.5) and max_depth (integer from 3 to 8) the same as before. For minimum child weight, I chose to extend the range from [1, 10] to [1, 20]. This is because each tree is separately trained until aggregated and should not overfit the data on which it is trained. Having a larger minimum child weight prevents each tree from overfitting. For subsample, I chose a uniform range [0.3, 1.0], which has a lower bound than the gradient boosting range. This is because each tree may be better subsampling a small portion of the huge training set to create simpler weak models. Finally, column samples by *tree* is now column samples by *node*, as each separate tree should have access to the full feature set, and then decide which features to subsample before splitting a node (or region). Since this subset should be smaller than the overall set of features, the range is uniformly distributed between [0.3, 0.9].

Overall, my decision to search the parameter spaces using Randomized Search Cross Validation was motivated by the extremely large size of the datasets, and the speed to train each model. Sklearn's RandomizedSearchCV is much quicker at searching a large parameter space than GridSearchCV, with negligible reduction in model quality over large datasets.

3.5 Model Selection and Comparison of Results

There was no pruning of model sets or model selection performed beyond the six models defined above. As previously mentioned, each model, for each rolling period, was trained on a 3-fold cross validation that searched over its parameter space. Since the Baseline model has no parameters, it iterated once, and the Baseline with PCA model iterated 10 times over its search space of PCA components. The final 4 models performed 80 iterations each for each rolling period, leading to a total of 2317 3-fold cross-validation iterations. The results showed that the data is extremely varied: the highest R^2 scores were only around 0.004, with the best random

forest model having a negative R^2 score. Figure 4 shows the log of the R^2 Score for each model over the rolling period intervals.

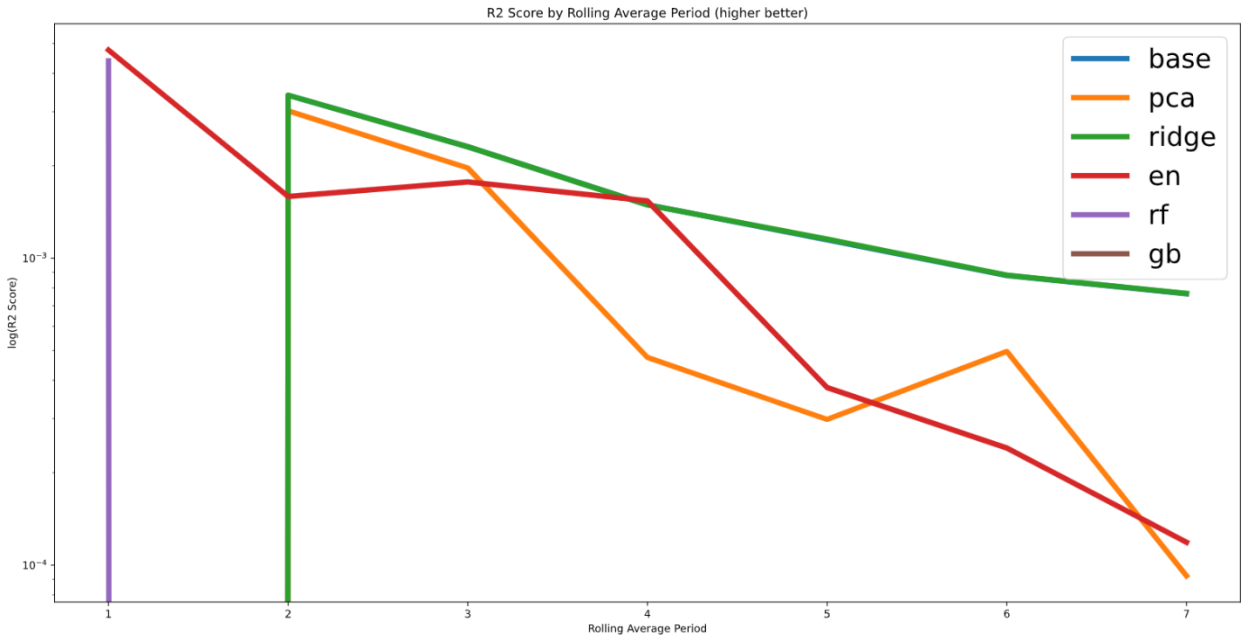

Figure 4: R2 score by rolling period for each model

For each model, the best performing rolling period model (scored by R^2) was then evaluated on the validation set (scored by MSE). These, again, led to remarkably similar values, all hovering around 0.025, which is extremely low but also entirely arbitrary and dataset specific. Nonetheless, every model performed better than the baseline estimator (though very marginally so in the case of Ridge Regression). The best performing model among those evaluated on the validation set was the random forest model, with an error improvement of **1.08e-4**.

Model	Rolling Period	R^2 Score (CV)	E_{val}	E_{test}
Baseline	1	-1.6921502590143414e+17		0.0221506
Params: **NONE**	2	0.0033946151290011515	0.0251670	
	3	0.0023076522480085884		
	4	0.0014927724496791583		
	5	0.0011478307476403178		
	6	0.000878393369216616		
	7	0.0007661755963946915		
Baseline with PCA	1	-6200529064142388.0		
Params: **PCA Components**: [2, 16]	2	0.003019498417768903	0.0251642	
	3	0.0019665475261048484		
	4	0.0004748127382357599		
	5	0.00029823296340543887		
	6	0.0004967630343982362		
	7	9.232473265108343e-05		
Ridge Regression	1	-1.517639282183242e+17		
	2	0.0033949581270365328	0.0251669	

Params: **PCA**: None, PCA(13 components) **Alpha**: [10^{-2}, 10]	3	0.002307002745229362		
	4	0.0014904248330300425		
	5	0.0011534022308694232		
	6	0.0008800085390628191		
	7	0.0007673715167845208		
Elastic Net Regression **PCA**: None, PCA(13 components) **Alpha**: [0.01, 10] (log) **L1 Ratio**: [0, 1]	1	0.004764657398258154	0.0251010	
	2	0.0015879658996526362		
	3	0.0017717192509192348		
	4	0.0015368782359166389		
	5	0.0003788090392313019		
	6	0.00024075220612129744		
	7	0.00011874449400961495		
Random Forest **PCA**: None, PCA(13 components) **Min. Child Weight**: [1, 20] **Gamma**: [0, 5] (normal, center 0.5) **Subsample**: [0.3, 1] **Col. Sample by Node**: [0.3, 0.9] **Max. Depth**: [3, 8]	1	0.004393273295041429	0.0250582	**0.0219962**
	2	-0.0010748846599584194		
	3	-0.0002480853390854169		
	4	-0.004550251555662539		
	5	-0.0012576569198678629		
	6	-0.0008423408445064062		
	7	-0.0017672018030502705		
Gradient Boosting **PCA**: None, PCA(13 components) **Learning Rate**: [0.01, 1] (log) **Min. Child Weight**: [1, 10] **Gamma**: [0, 5] (normal, center 0.5) **Subsample**: [0.5, 1] **Col. Sample by Tree**: [0.6, 1] **Max. Depth**: [3, 8]	1	-0.023488580555627376	0.0250772	
	2	-0.033911593262393645		
	3	-0.09343366822534478		
	4	-0.060785290783881916		
	5	-0.07072184886919335		
	6	-0.05787970074077579		
	7	-0.06298213318300316		

Table 4: Error rates for each model

The final parameters for each of the 6 best models are denoted in Table 5. These correspond to the highlighted entries in Table 4, with the green entry being the final model selected for testing. Note that the baseline model has a test error as well. This is just to gauge the relative strength of the final estimator and was not considered in the testing hypothesis set.

Model	Parameters
Baseline	NONE
Baseline with PCA	**PCA Components:** 12
Ridge Regression	**PCA:** None **Alpha:** 7.35901
Elastic Net Regression	**PCA:** None **Alpha:** 0.011496 **L1 Ratio:** 0.15376
Random Forest	**PCA:** PCA(13 components) **Min. Child Weight:** 19 **Gamma:** 0.551011 **Subsample:** 0.407939 **Col. Sample by Node:** 0.398620 **Max. Depth:** 5
Gradient Boosting	**PCA:** PCA(13 components) **Learning Rate:** 0.216496 **Min. Child Weight:** 1 **Gamma:** 1.07492 **Subsample:** 0.911210 **Col. Sample by Tree:** 0.964171 **Max. Depth:** 5

Table 5: Parameters for each model tested on Validation Set

4. Final Results and Interpretation

The final results, considering the relative closeness of the final model to the baseline estimator, prove to be inconclusive. While there is a level of improvement, it is not enough to prove any predictive capabilities of weather alone on the spread of coronavirus. There are simply too many variables to consider in the virus's spread, whether it is politics, adherence to disease prevention guidelines, socioeconomic makeup of a population, or even discrepancies in reporting, that could lead to noise that the current feature space is unable to fit. Even with the entire geographic and socioeconomic landscape of the US to use as a normalizer, different climate regions are roughly associated with their own cultures, political ideologies, and have different coronavirus restrictions in place that would affect growth rates. These can potentially correlate with weather patterns and need to be accounted for before a better estimate can be made.

$$E_{out}(h_g) \leq E_{Testing}(h_g) + \sqrt{\frac{1}{2N_{Testing}} \ln \frac{2M}{\delta}}$$

For probability $(1-\delta) = 0.95,\ M = 1,\ N_{Testing} = 60393,\ E_{Testing}(h_g) = 0.0219962$

$$E_{out}(h_g) \leq 0.0219962 + 0.0055263 = 0.0275225,\ \varepsilon_m = 0.0055263$$

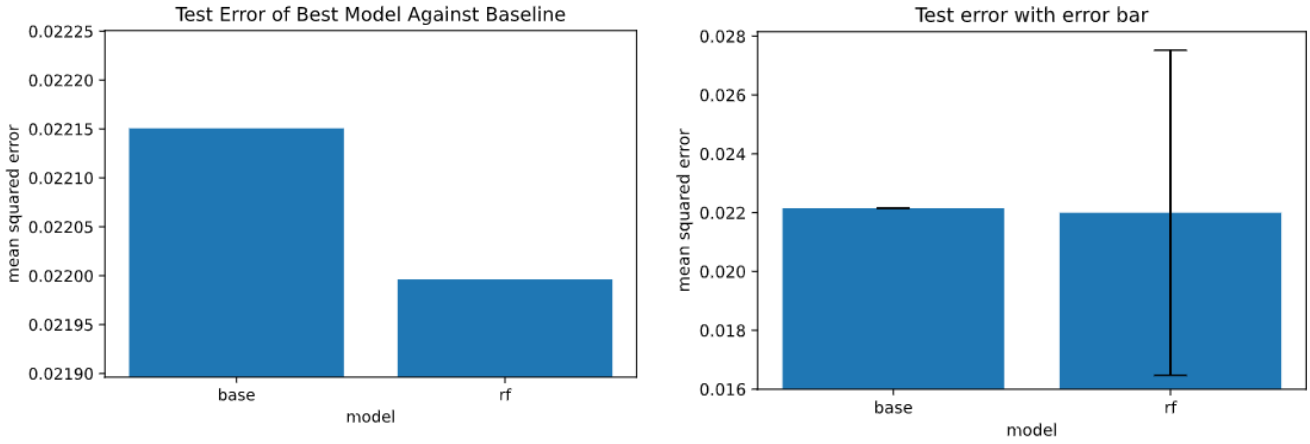

Figure 5: Final estimator (random forest model) test error against baseline test error

5. Contributions of Each Team Member – None

6. Summary and Conclusions

I would like to continue delving into the unknown factors that can exacerbate COVID-19 spread. As a first step, I need to be able to control for the multitude of factors with which weather itself is correlated, and then normalize my dataset for measuring case rates such that weather patterns are independent of those factors. The dataset I drew from also includes hundreds of

socioeconomic features (such as housing prices, unemployment, political affiliations, gender and race makeup of a population, etc.) that I discarded for the purposes of this project; too many features coupled with the already massive dataset would be impossible in scope to completely test. I also plan on automating my system in such a way that it could constantly learn from newer incoming data, and hopefully increase model accuracy and fit to the underlying truth.

7. References

[1] "MarketWatch," [Online]. Available: https://www.marketwatch.com/story/trump-says-coronavirus-could-be-thwarted-by-summer-heat-citing-dhs-study-2020-04-23.

[2] A. Briz-Redon and A. Serrano-Aroca, "The effect of climate on the spread of the COVID-19 pandemic: A review of findings, and statistical and modelling techniques," *Sage Journals,* vol. 44, no. 5, pp. 591-604, 2020.

[3] F. Mendez-Arriaga, "The temperature and regional climate effects on communitarian COVID-19 contagion in Mexico throughout phase 1," *Science of The Total Environment,* vol. 735, 2020.

[4] New York Times, [Online]. Available: https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-counties.csv.

[5] "NOAA," [Online]. Available: https://www.ncdc.noaa.gov/monitoring-references/maps/us-climate-regions.php.

[6] J. J. Davis. [Online]. Available: https://www.kaggle.com/johnjdavisiv/us-counties-covid19-weather-sociohealth-data .