Forecasting The Spread of COVID-19 with Enhanced Linear Autoregression

A Preprint

Ethan Mann[*]
California Institute of Technology
Computing + Mathematical Sciences Department
Pasadena, USA
emann@caltech.edu

Quentin Chevalier[*]
California Institute of Technology
Division of Engineering and Applied Science
Pasadena, USA
caltech@hawkspar.com

Geoffrey Magda[*]
California Institute of Technology
Division of Engineering and Applied Science
Pasadena, USA
magda.geoffrey@gmail.com

Matthew Gonzalgo[*]
California Institute of Technology
Computing + Mathematical Sciences Department
Pasadena, USA
gonzalgo@caltech.edu

September 24, 2020

Abstract

The COVID-19 pandemic is the focal point of global attention in 2020, as the virus has affected most of the world. Effectively responding to the pandemic continues to be a major and urgent issue; as of September 2020, there is still no widely accepted cure for the disease. To complement medical studies regarding the nature of the virus and provide decision-makers with tools to handle the crisis, Caltech offered the Spring 2020 CS156b course[2] to forecast the probability distribution of the number of COVID-19 related deaths. The challenge was held during April and May 2020, and the predictive goal was to model fatalities as reported by the New York Times (NYT) two weeks into the future on the county level, with predictions evaluated via pinball loss. This paper details our team's model submission, a linear autoregressive model with features for cases and fatalities.

1 Introduction

Our model for forecasting the spread of COVID-19 [1, 2] in the US has several key elements. At a high level, the model uses linear autoregression [3, 4] with engineered features for base predictions, applies growth bounds and blends various base predictors to create stable intermediate predictions, then clusters on validation error residuals to convert intermediate predictions into predicted probability distributions. With this approach composed of fundamental machine learning techniques, we achieved a high ranking in the CS156b challenge and developed a framework for researchers to make model improvements and enhance our model foundation.

Section 2 details the model design and training process. It provides data on model performance for two-week predictions during the CS156b challenge in Spring 2020 and over the subsequent months of Summer 2020.

Section 3 explores other aspects of the model. This analysis includes the model's sensitivity to noise in input data, model explainability in broad terms, and results of the model for prediction periods longer than two weeks.

[*]The authors listed are co-authors with equal contribution.

[2]We would like to acknowledge Professor Abu-Mostafa and the Caltech CS156b course staff for organizing the challenge, providing a pipeline of key datasets, and continuing research to develop the CS156 model. The resulting aggregate model from the continued project is displayed at http://cs156.caltech.edu/

2 Model Outline

2.1 Data Usage and Preprocessing

Given that the figure of merit was accuracy over NYT data[3], it was clear to us from the beginning that the NYT dataset would be treated as reference. Even if the NYT figures did not account for under-reporting at first, we still considered them as ground truth during training. Given our model, one could say that the most important property of the data is that NYT cases and fatalities are coupled and provide a consistent trend, even if the actual number of cases is indeed expected to be higher than the NYT data.

For the linear model, the primary strategy was to use time-dependent data for model features and clustering counties, and static data only for clustering counties when selecting model hyperparameters. This is intuitive, since static data features would be enveloped into the linear model constant; only time-dependent data can have an effect on training and model predictions, yet counties experiencing similar trends in COVID-19 are likely linked by various demographic, spatial, etc. types of static data similarities, and directly linked by time-dependent COVID metrics.

As we settled on only using NYT data to cluster counties for hyperparameter selection (discussed in more detail in “Training Strategy"), all static data sets were ignored, leaving mobility data [5] as the remaining time-dependent set for model features and further consideration. Ultimately, mobility data was not incorporated into the final model. As of model development in April/May, decreases in mobility at the beginning of the reported data (while cases and fatalities were both increasing) may have led to the model learning an inverse relationship between mobility and COVID-19 metrics. And as our model trains on all available dates, it seemed possible that our model as of Spring 2020 would experience this effect.

Thus, as of our final submission, our sole training dataset is that of the New York Times, with FIPS codes remapped for New York and Guam, and a minor interpolation for New York on 5/16/20 and 5/17/20, where the NYT data seemed arbitrarily frozen (validating over a few possible interpolations and permanently settling on the best-performing interpolation). The NYT data was used as is; no smoothing was performed. Clustering various counties based on their number of fatalities pilots the choice of hyperparameters, but the linear model is trained on a per-county basis. Notably, we did apply feature engineering via differencing the cumulative quantities to produce daily increase features. With respect to the linear model, this approach to data handling was fairly consistent throughout the project.

2.2 Model Description

Our final submission is a linear model that makes predictions using fatalities and cases data from the NYT dataset. For each county, we train two Ordinary Least Squares (OLS) autoregressors (one to predict cumulative fatalities and one to predict cumulative cases) based on features including cumulative cases/fatalities and daily increases in cases/fatalities. To predict, we use real/reported data as model input, generate a prediction for the first future day, and feed the prediction into the model per county to predict the next day, etc.

2.2.1 Growth Bounds

We reinforce realistic model behavior by preventing the model's predictions for cumulative values from decreasing (at all) or increasing (past a growth limit). The growth limit is computed with a validated parameter. There are two techniques for computing growth limits:

1. Method 1: the model inputs a parameter c and computes the largest increase in the training data (let this increase be x). It then prevents the model predictions from increasing by cx (i.e., applies a ceiling of cx).
2. Method 2: the model inputs a parameter d and looks at the largest increase over the past d days (let this increase be x_1) and compares it to the largest increase excluding the latest d days (let this increase be x_2). Then, the ratio x_1/x_2 is inputted as the parameter c to Method 1.

Our final model employs Method 2, and the growth limit is quite important to model stability, so consider the implications of our technique:

1. When $x = x_1$, the largest daily increase in the county's history occurred over the last d days. In this case, Method 2 prevents predicted increases more than $(x_1/x_2)x = (x_1/x_2)x_1$, i.e., the maximum increase from the last d days scaled up by a factor (x_1/x_2) representing an estimate of the rate of increase to expect.

[3]Data from The New York Times, based on reports from state and local health agencies.

2. When $x = x_2$, the largest increases occurred before the last d days. In this case, Method 2 prevents predicted increases more than $(x_1/x_2)x = (x_1/x_2)x_2 = x_1$, i.e., the maximum daily increase seen in the past d days.
3. Next, consider that when trained on noisy data (where a few days of no reporting could be registered as 0-values), the model could produce a growth bound with ratio $c = 0$ and prevent the final output from making a meaningful prediction. A solution to this is to provide a floor/ceiling for c for each tier used (e.g., $c \in (0.1, 1)$ for some models, $c \in (0.5, 2)$ for others, etc.) or to smooth the data in pre-processing and reduce noisy input. The first suggested solution (of placing a floor/ceiling on the computed values of c) is implemented in the model under a keyword "bounded" or "unbounded" within the growth parameter tuple representation, but we set the parameter at "unbounded" in our submission.
4. Finally, consider that the growth bound may be insufficient for determining growth restrictions on the model after a few days of predictions. One future suggestion is to have a dynamic growth bound that becomes more flexible over time or changes based on the model output, i.e., model predictions that saturate the growth bound result in a more flexible bound for the next prediction.

2.2.2 Model Hyperparameters

An instance of our general model class is tuned with the following hyperparameters:

- The fatalities and cases models each have 6 parameters that determine the OLS input features. For each model, these parameters include a time delay on cases and fatalities, along with the number of previous days (after applying a time delay) of the 4 feature types to include: cumulative cases, cumulative fatalities, daily increases in cases, and daily increases in fatalities.
- The fatalities and cases models each have growth bound parameters, as described above.
- The overall model has a minimum-fatalities parameter, i.e., specifying that any data points where cumulative fatalities is less than “minimum-fatalities" will not be included in training.

Increasing time delay or using more past days of data as features for a specific input vector has pros and cons. In one sense, extending model features further into the past is good for an autoregressive model since it allows predictions to be generated from real/reported data long after the initial predictions, and time delay itself is helpful to capture the delayed effect of cases on fatalities. However, time delay can prevent the model from having up-to-date data for its immediate predictions, although no data is lost, just delayed. Note also that adding more features/data to a single input vector will increase the number of model parameters and may result in overfitting (since as of Spring 2020, the number of training points is limited at a fixed 2 months regardless of features used).

2.2.3 Error Bars

Error bars are determined based on validation results (for more on validation, see “Training Strategy"). For each day of validation (7 days), we apply KMeans to place predictions (solely based on the value of predicted daily increase) into a cluster. For each cluster, we retrieve the validation errors for each FIPS county code in the cluster and sample a quantile distribution from these validation errors using NumPy. Early on, we found that KMeans clustered well for low magnitude predictions but for high magnitude predictions, adding additional clusters usually meant individual high-valued predictions would be placed in their own clusters (a reasonable side-effect of the loss function used in KMeans). As such, we ran validation on the error bar clustering to determine a suitable number of clusters and arrived at $n = 10$.

Moreover, as suggested above, some clusters contained very few points (especially for counties with large predictions), so to effectively sample from the validation error distribution over a cluster with 5 or fewer points, we pre-processed the validation errors by appending errors $\{x, 0.5x, -0.5x, -x\}$ for each error x in the cluster (and a base value of 0 for stability), then sampled from this processed list of errors. In accordance with reality, we apply a floor at 0 after adding error bars to our model prediction.

Any time there is a model prediction of 0, we do not apply error bars since this almost always means the county has not yet experienced or reported fatalities. Otherwise, for week 1, we directly use the validation error bar method to produce/apply error bars; for week 2, we apply the week 1 day 7 error bars. We apply a stretch-factor to extend the error bars by $E = 1.2$ (validated parameter) about the 50th quantile since error bars tend to be underestimated by our process.

2.2.4 Residuals and Over/underestimation

For weeks 1 and 2, we tried to tune the model based on its error residuals. For week 1, we considered training on the residuals, but shifting the prediction using error bars not centered at 0 (but rather centered at the 50th quantile of the error distribution) covered this well enough. For week 2, we deflate predictions linearly with time to avoid overestimation; after testing a few deflation functions, we settled on the deflation factor given by

$$\text{deflate(day, prediction)} = 1 \text{ if day} < 7 \text{ else } (1 - \text{day})/40 \tag{1}$$

2.2.5 Stability

We select a set of models to use per tier of counties via ensemble learning (applying different weights over each model). This is to reduce the variance in predictions. To circumvent dedicating additional data for validation and training a more extensive blending, we tried a few basic combinations of weights and settled on using 3 sets of model parameters per tier with [5, 3, 2] weights.

2.3 Training Strategy

We partitioned the NYT data into three chronological sets: training, validation, testing. Using standard methodology, we evaluated/compared between various models by training the model architectures and tuning through validation, then compared results (from the best model in validation, per architecture) on a test set. For the final model submission, we omitted the test set and trained on all available data.

2.3.1 Tiers

To determine hyperparameters for each county-specific model (as specified in "Model Description"), we partition the counties into several tiers. Our model assigns ID's to each cluster and then assigns counties to each ID, so any grouping scheme is possible. We focused on placing counties into intervals based on cumulative fatalities and cases, and our final model submission specifically placed counties into groups based on where cumulative fatalities fell over the intervals [5000, 1000, 250, 40, 10, 0].

The model contains code to allow for multiple separate tier assignments for counties, run validation over each set of tier splits, and form the prediction for a county by taking an average over multiple ensembled models (an extension on already averaging multiple models per tier, as mentioned in “Stability").

2.3.2 Grid Search, Validation, Loss Functions, and Ensembling

Given a tier of counties, we perform grid search over ranges of parameters for each model setting (with ranges customized for high-fatality vs. medium-fatality vs. low-fatality counties). For a given set of parameters, we compute a validation score by:

1. Training the customized cases and fatalities OLS models on the training data with a loss function of Mean Squared Error (MSE) on cumulative predictions
2. Making model predictions on daily increases over a 7 day validation period
3. Computing the MSE of the predicted daily increases against real daily increases

We tried switching between MAE vs. MSE for validation error (and this can be adjusted for any specific tier in validation) but it seemed MSE was better suited for performance on the testing set.

Within a tier assignment and using the [5, 3, 2] weights (see “Stability"), we pick the best model from validation, then find the best model to add to the ensemble with relative weighting, and then find a third model to complement the pair. The intention of this is to consider $O(3n)$ instead of $O(n^3)$ ensembles/trios of models (given some n possible model parameters).

2.3.3 Evaluation and Comparison

After we select the 3 models to ensemble based on validation, we retrain the tier's models on the training/validation data and score them against the testing data, on pinball loss [6], to confirm that the model can generalize. We used the testing score to advise various decisions throughout the project (i.e., the best tier splits to use, selection of constants such as $N = 10$ clusters in KMeans for error bars, etc.).

2.3.4 Careful Handling of Data

This training/validation process may have led to very minor "data contamination." For validation, we exclusively train on training data to predict validation data but we place counties into tiers based on data from the end of the validation period. This is important because counties starting to experience COVID-19 may underperform in validation, unless we apply recent data to correctly assign tiers. When we tried to eliminate this potential internal contamination by creating tiers solely based on training data, then validating and making test predictions, the testing error increased, so we kept the original technique in the final model.

2.3.5 Successful Tier Approaches and Updating the Tier System over Time

One interesting pattern was a pyramid structure that emerged in the sizes of tiers employed in the model. The final model in May 2020 had groupings of sizes [1, 14, 52, 175, 386, 2310].

Over time, counties accumulate more cumulative fatalities (a natural effect of time), so given a fixed set of cumulative fatality intervals used to assign tiers, the lower intervals of fatalities will gradually decrease in size (size being the number of FIPS they contain) while the higher intervals of fatalities will increase in size. Accordingly, we updated the county tier splits over the first half of the project to maintain the ideal balanced (“pyramid") structure, where higher intervals contain fewer FIPS and lower intervals contain more FIPS. However, we did not perform any tier split updates over the final weeks. Moving forward, it may be useful to add a few more tiers (or at least update the splits to reflect increases from all counties), with the following guidelines:

- Keeping New York City in its own tier has improved evaluation results (since New York City has a large influence on overall pinball loss). As such, the tier of size 1 is a tier with New York City alone.
- The tier of size 14 blends all of the highly-impacted counties together in the selection of hyperparameters, so future work in splitting up the second-highest tier of counties could improve the "blurring" of Boston vs. Chicago vs. Los Angeles.
- Training with case tiers or case/fatality hybrid tiers did not do as well as fatality tiers alone. This is probably since fatalities data tracks most closely with the target predictions.
- The boundary between the lowest two clusters is somewhat sensitive to changes, so adding tiers may be most effective when splitting tiers other than the lowest tier.

2.4 Model Robustness

Relative to other models in Spring 2020’s CS156b, our model performed its best during the first and second course checkpoints. Nevertheless, the model performance was strong throughout the duration of the challenge. Here is a quantitative summary of model robustness:

2.4.1 Spring 2020 Performance

Over an evaluation period from 5/25/20 to 6/3/20, our final model had 0.1032 pinball loss and 1.5908 Root Mean Squared Error (RMSE). For reference, sample predictions of all 0’s had 0.1547 pinball loss and 2.2089 RMSE.

See Figure 1 for plots of the 1-day model scores (pinball loss and RMSE) against the Zero predictions as a baseline.

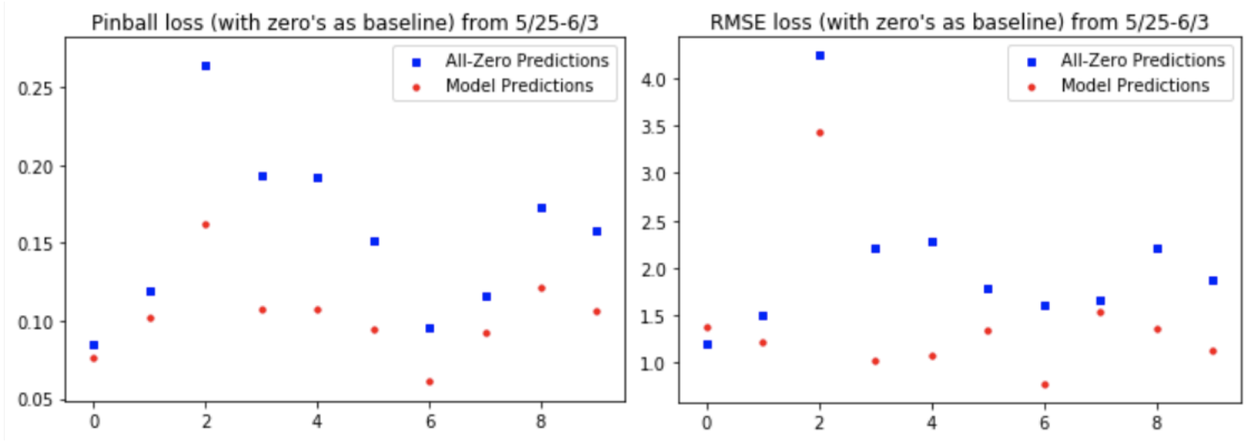

Figure 1: Daily Pinball loss and RMSE scores

Additionally, to see how the model holds over time, consider Figure 2 for a snapshot of the model's residual errors toward the beginning of the evaluation period (5/27/20) and toward the end of the evaluation period (6/3/20).

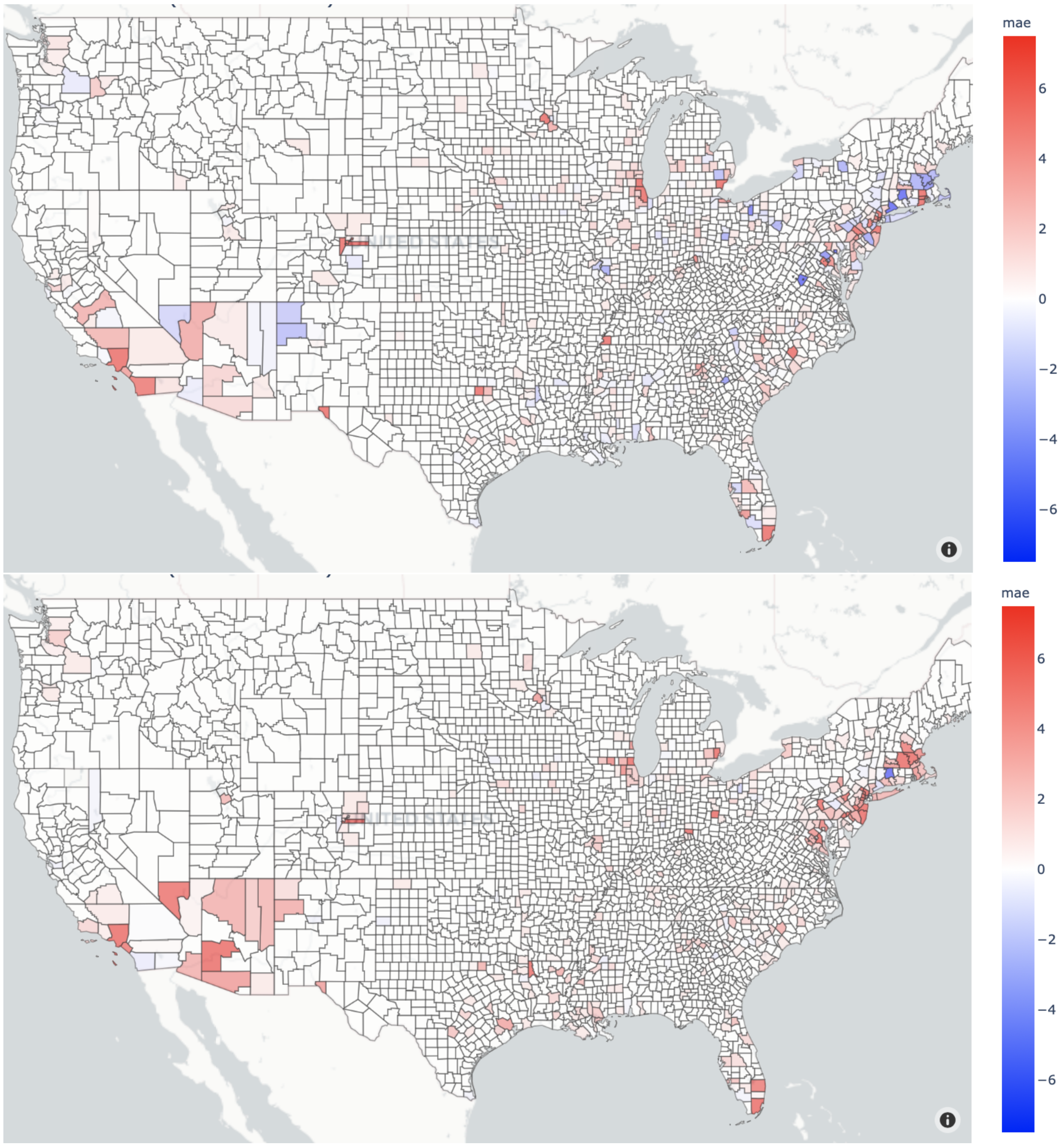

Figure 2: The top plot is error residuals on 5/27/20. The bottom plot is error residuals on 6/3/20. Residuals are (real value) minus (50th quantile prediction); red is underprediction.

Toward the beginning of the evaluation period, there is a mixture of overprediction and underprediction. But after 10 days, the model appears to underpredict over most counties. This could be because of the growth bound (which may have placed a tight upper bound on model predictions) or deflation function (which may have been more useful while most counties were experiencing a deceleration in COVID-19 fatalities during previous weeks). However, in the face of increased mobility around the end of May 2020, it is understandable that we were underestimating across the US during this time period.

2.4.2 Counties with Greatest Error

We examined counties with the largest absolute and percentage errors. The largest absolute errors are from large population counties, where the model may be 1) grouping too many high-fatality counties together to appropriately capture individual county changes in validation, 2) unable to process a county with a resurgence of COVID-19 (where the growth bound prevents the model from making sufficiently large predictions), or 3) experiencing similar underestimation over all counties but with a more magnified effect on counties with larger, more variable reported increases.

Mean absolute error	Location	Percent error
25.208	Cook, Illinois	52.958
22.028	New York City, New York	46.571
17.600	Providence, Rhode Island	100.000
13.740	Los Angeles, California	35.688
8.909	Wayne, Michigan	74.242
8.758	Middlesex, Massachusetts	55.430
8.702	Hartford, Connecticut	102.376
8.438	Essex, Massachusetts	70.317
7.978	Miami-Dade, Florida	79.780
6.295	Worcester, Massachusetts	50.360

Table 1: Top 10 Absolute Errors averaged from 5/25/20 to 6/3/20 (listed with location and percentage error)

The largest percentage errors seem to be from lower population counties, where the mean absolute error is low, but so are the target values, so it is feasible to have percent errors as large as 400% or 500%.

Percent error	Location	Mean absolute error
524	Hancock, Georgia	2.096
399	St. Joseph, Indiana	0.399
387	Delaware, Ohio	0.387
375	Harford, Maryland	1.125
340	Outagamie, Wisconsin	0.340
292	Leake, Mississippi	0.583
277	Deaf Smith, Texas	0.277
273	Beaver, Pennsylvania	0.546
253	Collier, Florida	1.013
250	Sumner, Tennessee	0.250

Table 2: Top 10 Percentage Errors averaged from 5/25/20 to 6/3/20 (listed with location and absolute error)

2.5 Summer 2020 Performance

Additional performance data from Summer 2020 indicates the model offers a pinball loss and RMSE improvement over the all-zero prediction baseline for all time periods except the 8/1/20-8/15/20 window (which only has a rise in RMSE).

Date range	Model Pinball loss	Model RMSE	Zero's Pinball loss	Zero's RMSE
6/15/20 - 7/1/20	0.0917	2.8518	0.1151	2.9675
7/1/20 - 7/15/20	0.0883	3.0362	0.1069	3.2825
7/15/20 - 8/1/20	0.1282	2.7038	0.1649	3.0958
8/1/20 - 8/15/20	0.1621	4.2137	0.1777	3.2686
8/15/20 - 9/1/20	0.1170	1.2827	0.1517	1.8143
9/1/20 - 9/15/20	0.1041	1.2002	0.1300	1.5616

Table 3: 14-day prediction average loss from Summer 2020

3 Model Utility

3.1 Model Sensitivity

Our model was retrained with perturbed input in order to quantify sensitivity to changes in death and cases data. Perturbation was done by multiplication by a uniform random factor $r \in [1-\epsilon; 1+\epsilon]$. A new factor was chosen for every row of the data for the last T days. In order to take advantage of the latest data, the most recent version of the model was run for a two week prediction ending on the last available day of data when this section was written (6/1/20).

This translated into the following code:

```
def tweak_data(df, cols, last_days, eps, last_training_day):
   msk = pd.to_datetime(df['date']) > last_training_day-timedelta(days=last_days)
   for col in cols:
      df.loc[msk,col] = df.loc[msk,col]*(1+eps*(1-2*np.random.random(np.sum(msk))))
   return df
```

The resulting graphs are presented in Figures 3 and 4 for fatalities and cases perturbation, respectively. One can see that pinball loss grows linearly with perturbation amplitude, at a faster rate as T increases. Notice also that variability around the trend increases with T, which leads to an decrease in the regression factor R^2. This behavior remains true at very high relative perturbations; the largest amplitude corresponds to a 30% change in the data. RMSE loss is harder to interpret, especially given a major outlier in the five days perturbation, mostly explained by very large RMSE error on two different days.

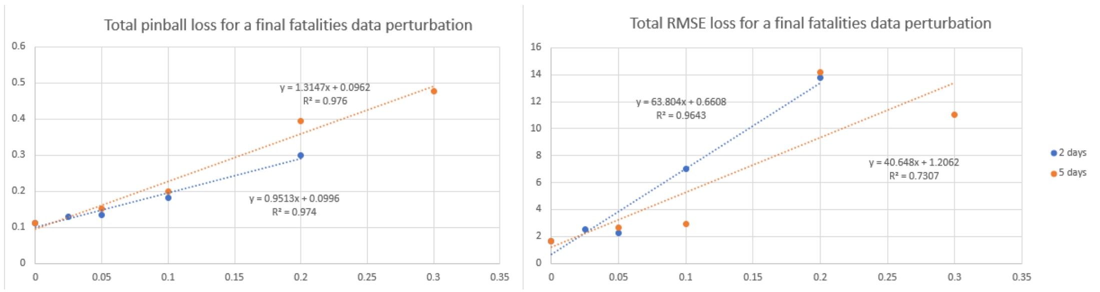

Figure 3: Model losses with respect to fatalities perturbation amplitude ϵ for $T \in \{2, 5\}$

Figure 4 shows how shifting cases induces a less linear behavior for either loss than fatalities (lower R^2) and a much lower influence on either loss (lower slope than fatalities).

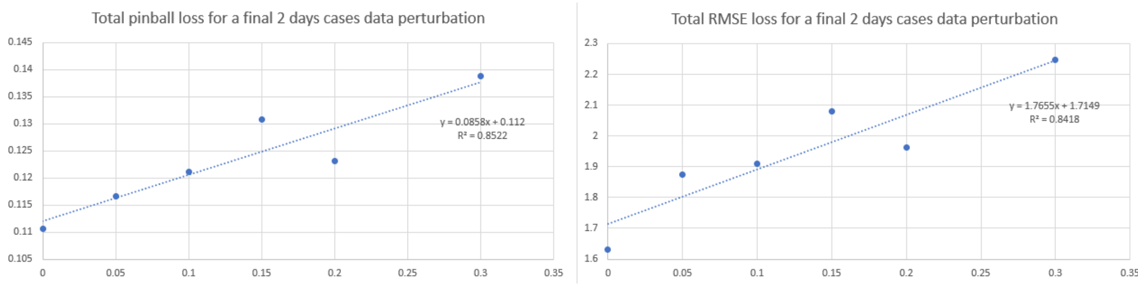

Figure 4: Model losses with respect to cases perturbation amplitude ϵ for $T = 2$

One may quantify the sensitivity of our model to random noise with respect to these figures and determine expected error magnitude.

3.2 Model Explainability

In our model we only use the NYT data (death and cases). There is a clear correlation between cases and fatalities. This can be used to catch long term trends that we cannot predict from the fatalities data alone. For instance, take two counties that reported 70 deaths on 4/1/20: County 34013 and County 36103 (identified by their FIPS codes).

County 34013 has a much more significant increase rate for deaths (30% per day increase over the 10 previous days, compared to 20% for 36103). However, the number of cases is far more significant for county 36103 (7605) than for county 34013 (2262). Two weeks later, county 36013 has more reported deaths (663 versus 590). Incorporating the data source of reported cases improves model predictions and the ability to explain scenarios like this; case figures are more statistically sound at the beginning of the spread of COVID-19 than fatalities (because of low numbers of reported deaths).

In general, even if two counties have different death rates (and often they do), the rate of increase of cases is a good replacement (at least partially) of the death data when the latter is too low to be statistically useful. As such, including cases in the model is definitely helping on average.

3.3 Longer Time Horizon Predictions

During Spring 2020, we used the model to make predictions over four month-long periods (where the first day listed is the last day in training): April 1 - May 1, April 10 - May 10, April 20 - May 20, and April 30 - May 30.

This was using the model with no adjustments to the [5000, 1000, 250, 40, 10, 0] tier splits. The deflation function was modified to prevent additional deflation past that of 14 days.

Date range	Model Pinball loss	Model RMSE	Zero's Pinball loss	Zero's RMSE
4/1/20 - 5/1/20	0.2937	10.8055	0.3181	11.1527
4/10/20 - 5/10/20	0.2258	6.0363	0.3122	8.9003
4/20/20 - 5/20/20	0.1870	3.8533	0.2751	5.5257
4/30/20 - 5/30/20	0.1660	2.9234	0.2179	3.5943

Table 4: Long range prediction losses from Spring 2020

We also have data from model performance over various month-long periods from Summer 2020. Performance data is summarized with testing start dates from June to August, with no modifications to the deflation function.

Date range	Model Pinball loss	Model RMSE	Zero's Pinball loss	Zero's RMSE
6/15/20 - 7/15/20	0.0917	2.9654	0.1109	3.1283
7/1/20 - 8/1/20	0.1208	2.3656	0.1373	2.4520
7/15/20 - 8/15/20	0.1387	2.8807	0.1715	3.1853
8/1/20 - 9/1/20	0.1417	2.8411	0.1625	2.6305
8/15/20 - 9/15/20	0.1148	1.2923	0.1397	1.6684

Table 5: Long range prediction losses from Summer 2020

The model's pinball loss makes strong improvements on the baseline zero predictions. With the exception of the 8/1/20-9/1/20 predictions (the same period as the previously mentioned spike in 14-day RMSE), the model RMSE always outperforms the baseline.

4 Conclusion

This model offers a statistical approach to modeling the spread of COVID-19. By utilizing fundamental machine learning methods, our team developed an autoregressive linear model that generates competitive predictions, with performance data provided for 14 day and 28 day predictions from Spring and Summer 2020. This work and the suggestions provided in this report can be applied to further research in machine learning for COVID-19 forecasting.

References

[1] Na Zhu, Dingyu Zhang, Wenling Wang, Xingwang Li, Bo Yang, Jingdong Song, Xiang Zhao, Baoying Huang, Weifeng Shi, Roujian Lu, Peihua Niu, Faxian Zhan, Xuejun Ma, Dayan Wang, Wenbo Xu, Guizhen Wu, George F. Gao, and Wenjie Tan. A novel coronavirus from patients with pneumonia in china, 2019. *New England Journal of Medicine*, 382(8):727–733, 2020. PMID: 31978945.

[2] Catrin Sohrabi, Zaid Alsafi, Niamh O'Neill, Mehdi Khan, Ahmed Kerwan, Ahmed Al-Jabir, Christos Iosifidis, and Riaz Agha. World health organization declares global emergency: A review of the 2019 novel coronavirus (covid-19). *International Journal of Surgery*, 76:71 – 76, 2020.

[3] R. Adhikari and R. Agrawal. An introductory study on time series modeling and forecasting. *ArXiv*, abs/1302.6613, 2013.

[4] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo. Arima models to predict next-day electricity prices. *IEEE Transactions on Power Systems*, 18(3):1014–1020, 2003.

[5] Michael S. Warren and Samuel W. Skillman. Mobility changes in response to covid-19, 2020.

[6] Ingo Steinwart and Andreas Christmann. Estimating conditional quantiles with the help of the pinball loss. *Bernoulli*, 17(1):211–225, 02 2011.