

# Machine Learning Engineer Nanodegree

---

## Capstone Project - DengAI Competition on DrivenData

---

Andy Gaworecki

December 20, 2018

### I. Definition

---

#### Project Overview

Dengue is a mosquito-transmitted viral disease affecting millions of people every year, including U.S. travelers and residents of the tropical regions of the U.S. like Puerto Rico. Experts estimate that around 390 million dengue infections occur worldwide each year, including about 500,000 severe cases – mostly children -- requiring hospitalization. Case counts are climbing as the disease moves into new areas. Dengue is now endemic in more than 100 countries and several U.S. territories. In endemic areas, major epidemics occur roughly every 3-5 years overwhelming medical services, so anticipating epidemics has the potential to save lives. The risk to U.S. travelers is on the rise, and in recent years, local dengue outbreaks have struck the continental United States where the *Aedes* vector mosquitoes are endemic.<sup>1</sup>

Accurate dengue predictions would help public health workers, like Johansson, and people around the world take steps to reduce the impact of these epidemics. But predicting dengue is a hefty task that calls for the consolidation of different data sets on disease incidence, weather, and the environment.<sup>1</sup>

This project is a competition from DrivenData.co and the purpose is to predict the number of reported cases of dengue fever for two tropical cities; San Juan, Puerto Rico and Iquitos, Peru. The data for model training in this competition comes from multiple sources aimed at supporting the [Predict the Next Pandemic Initiative](#). Dengue surveillance data is provided by the U.S. Centers for Disease Control and prevention, as well as the Department of Defense's Naval Medical Research Unit 6 and the Armed Forces Health Surveillance Center, in collaboration with the Peruvian government and U.S. universities. Environmental and climate data is provided by the National Oceanic and Atmospheric Administration (NOAA), an agency of the U.S. Department of Commerce.

In previous attempts to algorithmically predict number of dengue fever cases there has already been a great deal of research done, due primarily to the fact that in tropical parts of the world dengue fever is a serious health threat and economic burden. It seems that most every type of machine learning algorithm has been used to model cases of dengue fever. Many types of regression models have been tested including Support Vector Regression (SVR), Negative Binomial Regression (NBR), Gradient Boosted

<sup>1</sup> “Back to the Future: Using Historical Dengue Data to Predict the Next Epidemic”

<<https://obamawhitehouse.archives.gov/blog/2015/06/05/back-future-using-historical-dengue-data-predict-next-epidemic>> (2015)

Regression Tree Algorithm and LASSO linear regression. The most interesting paper on these models, for me, is titled [“Developing a dengue forecast model using machine learning: A case study in China”](#), published on the National Center for Biotechnology Information (NCBI) website and compares many types of regression models in predicting weekly dengue cases in Baidu, China using features of climate factors (mean temperature, relative humidity and rainfall) as features and an RMSE metric. This team concluded that the SVR model consistently produced the smallest prediction error rates.

Several papers investigate the use of approximate entropy algorithms for dengue case prediction and use the sensitivity of predictions as an error metric. Another [paper](#) explores the use of Artificial Neural Networks, which look like simple sully connected layers but the architecture is not specified, to make dengue case predictions in Mexico and San Juan, Puerto Rico (one the the challenge cities). This research created four different models for different subsets of the population and different locations and uses F-measure and ROC scores as error metric and achieves. They consistently achieved ROC and FM scores above 0.7.

Finally there is another paper from NCBI titled [“Ensemble Method For Dengue Prediction”](#). This model was built as part of the 2015 NOAA Dengue Challenge to predict dengue cases for specific periods in the dengue transmission season for both of the cities of interest in this project. They used an ensemble of three different statistical methodologies; Holt-Winters, Method of Analogues, and Historical models. They build three different ensemble models to predict three different aspects of the dengue transmission season; the number of cases in the peak week, identification of the peak week, and the total of cases in the season. MAE and RMSE were used to measure the error for different types of predictions.

## Problem Statement

The goal is to predict the **total\_cases** (of Dengue fever) label for each **(city, year, weekofyear)** in the test set. The training dataset contains about 18 years of data for San Juan and 10 years of data for Iquitos. The two cities, San Juan and Iquitos, have test data for each city spanning 5 and 3 years respectively. Only one submission will be made that contains predictions for both cities for each year and weekofyear in the test dataset. The tasks involved are:

1. Download and preprocess the training dataset from DrivenData.co DengAI competition
2. Engineer new features for the training dataset to better capture the environmental conditions favorable to mosquito population growth and hopefully disease spread by infection
3. Choose a method to select the most predictive features (simple correlation, PCA, or Sklearn feature selector)
4. Investigate Negative Binomial Regression models and choose best learning rate
5. Fit the model to the selected features
6. Make predictions for total cases of Dengue fever for each row of features in the test dataset
7. Evaluate prediction results with MAE error metric (via DrivenData competition) and visualizations

## Metrics

The metric for this project will be the one prescribed by the DrivenData competition, mean absolute error (MAE). This error metric is especially desirable because it is easily interpretable as a measure of average variance across all the model predictions for number of reported Dengue fever cases.

This is also a common metric for regression and time series predictions.

$$\text{MAE} = \frac{\sum_{i=1}^n |y_i - x_i|}{n} = \frac{\sum_{i=1}^n |e_i|}{n}.$$

## II. Analysis

---

### Data Exploration

The training data for this competition is contained in two files, 'dengue\_features\_train.csv' and 'dengue\_labels\_train.csv'. The features files contains 1456 rows (936 for San Juan and 520 for Iquitos) and each row is a weekly observation. This is about 18 years of data for San Juan and 10 years of data for Iquitos. There are 24 columns (features) in the features dataset as described below.

#### City and date indicators

- `city` – City abbreviations: `sj` for San Juan and `iq` for Iquitos - the only categorical variable
- `year` – Calendar year
- `weekofyear` – Week number of the calendar year
- `week_start_date` – Date given in yyyy-mm-dd format

#### NOAA's GHCN daily climate data weather station measurements

- `station_max_temp_c` – Maximum temperature
- `station_min_temp_c` – Minimum temperature
- `station_avg_temp_c` – Average temperature
- `station_precip_mm` – Total precipitation
- `station_diur_temp_rng_c` – Diurnal temperature range

#### PERSIANN satellite precipitation measurements (0.25x0.25 degree scale)

- `precipitation_amt_mm` – Total precipitation

#### NOAA's NCEP Climate Forecast System Reanalysis measurements (0.5x0.5 degree scale)

- `reanalysis_sat_precip_amt_mm` – Total precipitation
- `reanalysis_dew_point_temp_k` – Mean dew point temperature
- `reanalysis_air_temp_k` – Mean air temperature
- `reanalysis_relative_humidity_percent` – Mean relative humidity

- `reanalysis_specific_humidity_g_per_kg` – Mean specific humidity
- `reanalysis_precip_amt_kg_per_m2` – Total precipitation
- `reanalysis_max_air_temp_k` – Maximum air temperature
- `reanalysis_min_air_temp_k` – Minimum air temperature
- `reanalysis_avg_temp_k` – Average air temperature
- `reanalysis_tdtr_k` – Diurnal temperature range

Satellite vegetation - Normalized difference vegetation index (NDVI) - NOAA's [CDR Normalized Difference Vegetation Index measurements](#) (0.5x0.5 degree scale)

- `ndvi_se` – Pixel southeast of city centroid
- `ndvi_sw` – Pixel southwest of city centroid
- `ndvi_ne` – Pixel northeast of city centroid
- `ndvi_nw` – Pixel northwest of city centroid

I will likely drop the columns `['ndvi_ne', 'ndvi_nw']` to avoid finding a method to fill in about 13% of those observations and because this feature does not correlate with number of total\_cases. Less than 3% of values are missing for every remaining feature in this dataset (*figure 1*). The other missing values should be easily imputed because there are only a few instances of NaN values for each column.

There are many outliers in the features that measure precipitation and vegetation but the rest of the features have very few outliers (*figure 2*). I have decided to leave the outliers in the dataset because they may correlate with seasonal fluctuations in environmental conditions that correlate with total\_cases of dengue fever. We can also see that for different features there are significantly more outliers for one city than the other.

Number of NaN values for each feature:

|                                       |     |
|---------------------------------------|-----|
| city                                  | 0   |
| year                                  | 0   |
| weekofyear                            | 0   |
| week_start_date                       | 0   |
| ndvi_ne                               | 194 |
| ndvi_nw                               | 52  |
| ndvi_se                               | 22  |
| ndvi_sw                               | 22  |
| precipitation_amt_mm                  | 13  |
| reanalysis_air_temp_k                 | 10  |
| reanalysis_avg_temp_k                 | 10  |
| reanalysis_dew_point_temp_k           | 10  |
| reanalysis_max_air_temp_k             | 10  |
| reanalysis_min_air_temp_k             | 10  |
| reanalysis_precip_amt_kg_per_m2       | 10  |
| reanalysis_relative_humidity_percent  | 10  |
| reanalysis_sat_precip_amt_mm          | 13  |
| reanalysis_specific_humidity_g_per_kg | 10  |
| reanalysis_tdtr_k                     | 10  |
| station_avg_temp_c                    | 43  |
| station_diur_temp_rng_c               | 43  |
| station_max_temp_c                    | 20  |
| station_min_temp_c                    | 14  |
| station_precip_mm                     | 22  |

Figure 1 - Number of missing values for each column of training features

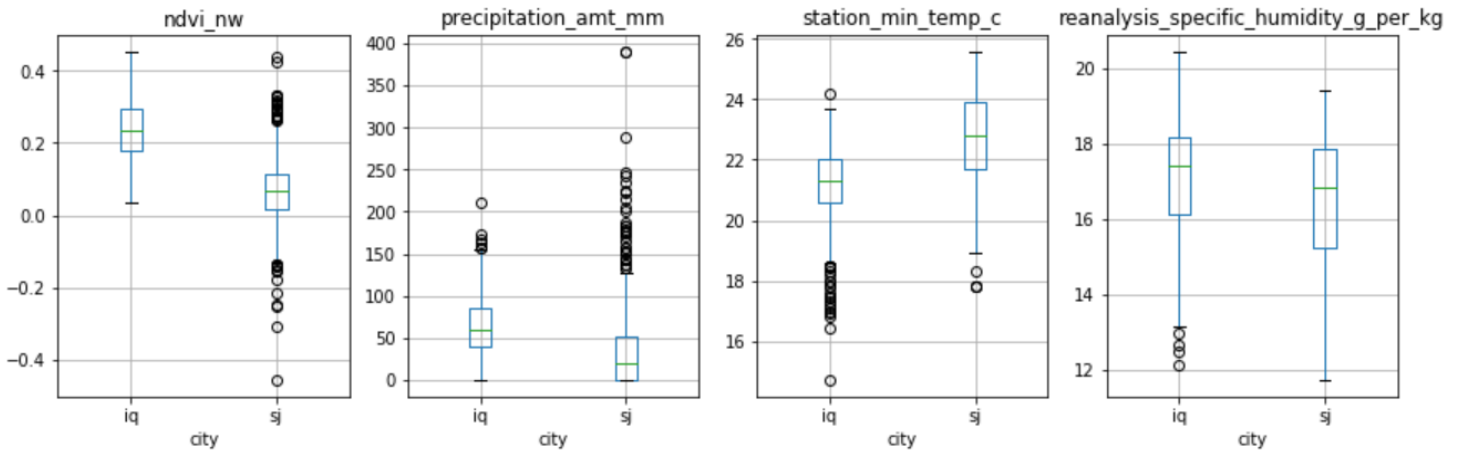


Figure 2 - Boxplot Examples of different features

# Exploratory Visualization

One of my initial investigations into this dataset was to explore the target variable, total\_cases of dengue fever. Along with some summary statistics I examined the distribution of total\_cases for both cities in the dataset with a histogram and a time series plot (figure 3, figure 4). From these histograms we can see that for both cities the number of total\_cases has an extremely heavy right skew with a mean much less than the variance. This is significant because it helps determine what type of statistical model would best fit this target variable. From the times series plots we can see that there appears to be a seasonal increase in the number of total\_cases, roughly annual, but there is also a good deal of variance to this cycle and some years are much higher than others. I will not model this data as a time series but that is something to consider for future analysis.

By comparing the plots for the two cities we can see that there is quite a bit more training data for San Juan. Overall San Juan tends to have more cases of dengue fever as shown in both the frequency distribution on the histograms and the largest peaks on the time series plots.

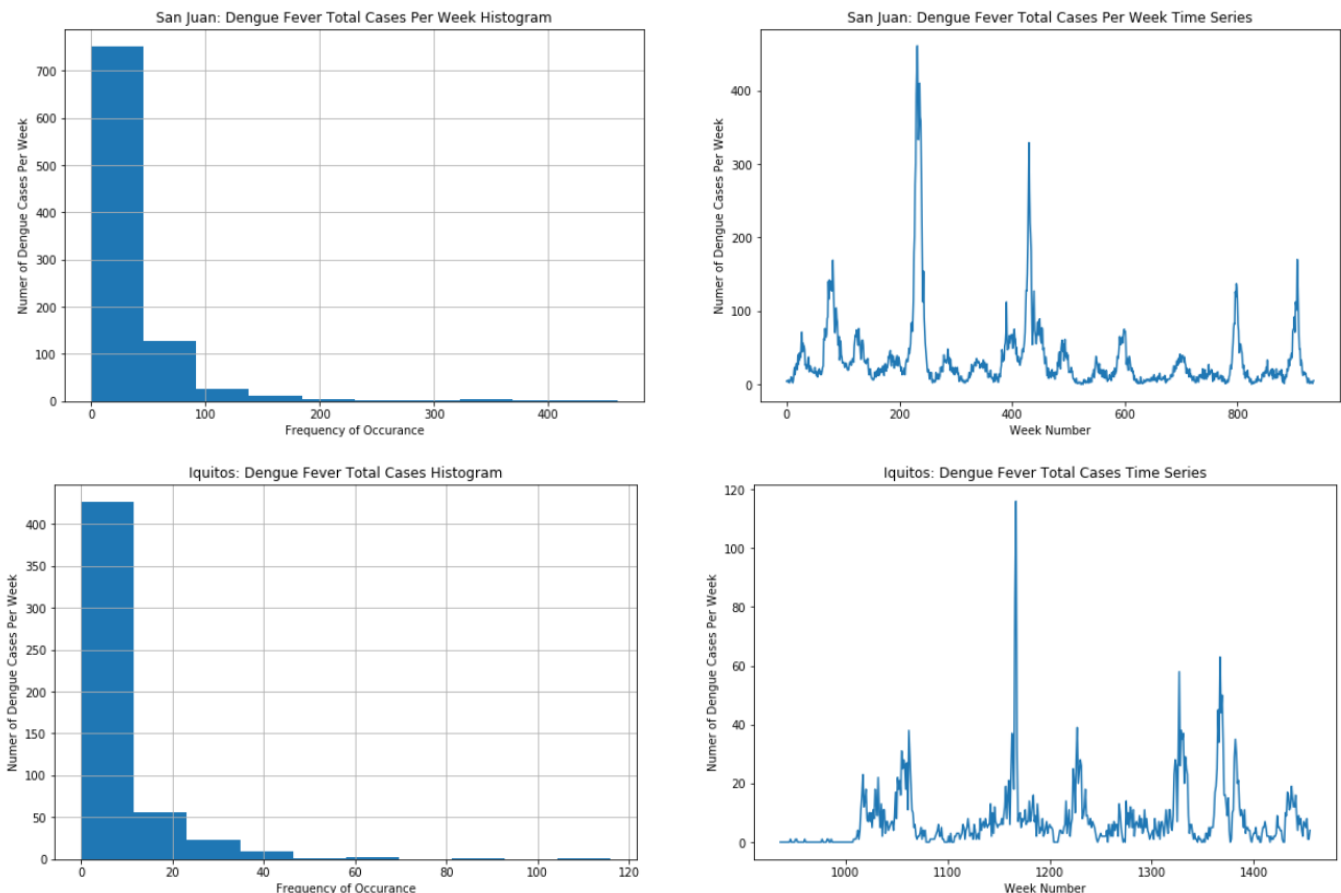


Figure 3 – Plots of total\_cases of dengue fever for cities of San Juan, Puerto Rico (top) and Iquitos, Peru (bottom)

## Algorithms and Techniques

The variables in the features dataset are a variety of environmental features and none of them have correlate well with total\_cases of dengue fever (figure 5). After pre-processing the features, I plan to do some feature engineering consisting of lagging

the features to account for the lag from mosquito life cycles and the incubation period of dengue fever (~ 6 week lag). I will then have over 100 features and will need to do some feature selection to choose the best features for modeling total\_cases. I am not sure which feature selection technique will work best in this case but I plan to experiment with manual feature selection based on correlation, PCA, Lasso regression, and *sklearn.feature\_selection.SelectKBest*. This will reduce the number of features in the final models and help reduce the noise from less important features.

I plan to use a Negative Binomial Regression (NBR) model to predict the number of weekly total\_cases of dengue fever because this target variable distribution fits thie Negative Binomial Distribution quite well mainly base on the characteristics:

- Modeling a discrete count (non-negative integer) of event in a specific time period
- Variance is significantly larger than the mean

Negative binomial regression is similar to standard multiple regression except that the dependent variable (Y or total\_cases) is an observed count (positive integer) that follows the negative binomial distribution. The NBR equation takes the form:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k + \epsilon$$

The beta ( $\beta$ ) terms represent the regression coefficient for each input climate feature (x) to the NBR. The algorithm will estimate the regression coefficient for each feature using Maximum Likelihood Estimation (MLE). Intuitively, MLE will determine beta coefficient values for each feature that make the Y values the most statistically probable, for all the sum of observations in the training dataset.

I plan to split the features dataset into separate datasets for each city to account for any specific differences in the dengue fever total\_count pattern between the two cities and build a separate NBR model for each city. I will split each city specific dataset into separate training and validation



datasets with a validation dataset the approximate size of the test dataset (about 20-30% of the features dataset) for each city. I also plan to loop through different learning rates (alpha) during the training and validation of these models to determine the most effective learning rate parameter.

## Benchmark

DrivenData has built a benchmark model using a [Negative Binomial Regression](#) algorithm with naive parameters and assumptions and no feature engineering or selection. This model achieved an MAE result of 25.8173 and my goal is to create a model with an MAE of less than 20.0.

## III. Methodology

---

### Data Preprocessing

Preprocessing Steps to take:

1. Capture with the month value from week\_start\_date feature and 'weekofyear' feature.
  - Month and weekofyear are proxy for seasonality of dengue fever total\_cases
2. Capture is\_sj data and the weekofyear data as integer variables for later use.
3. Delete all the time series features (year, weekofyear, week\_start\_date).
4. Interpolate to fill all missing data values. Most features have very few NaNs to be interpolated but a couple features have a large number. This may not be a factor because these features seem to have very little power to predict total\_cases and may be dropped instead of imputed.
  - Zero NaN values are a requirement for most machine learning algorithms
5. Create a function to create six lagged features for each variable remaining in the features dataframe. Each lagged feature will represent the original feature 1-6 weeks in the past.
  - Backfill NaN values at the front end of each lagged dataframe with either 1 or the mean value for that feature.
6. Add back in captured features of month and weekofyear.
7. Split the features dataframe into separate dataframe for each city, sj\_features and iq\_features respectively
8. Add target feature, total\_counts, column back to each features dataframe. This is required for the Negative Binomial Regression model.
9. Standardize all features with MinMaxScaler or StandardScaler from Scikit-Learn
  - Required by Lasso Regression
10. Split the two features dataframes into train and validation sets for cross-validation purposes.
11. Build a function preprocess\_data to perform all the preprocessing steps at once.

### Implementation

After preprocessing my model consists of three main parts; creating a Lasso Regularization model for each features dataframe and selecting the best dimensions from each dataframe to use in the NBR model, running cross validation on two different NBR models to determine the best features and parameters for modeling total\_cases, and finally making predictions on the features\_test

dataframe using the two trained NBR models and submitting the predictions to DrivenData for scoring.

Because I used Lasso regularization particularly for feature reduction my model involved cross validation on the alpha parameter and another parameter, I created which is the threshold value for the coefficient of the Lasso regression in order for a feature to be kept in the final model. I know that I should have done this cross validation in a more systematic, pythonic way but I just used a lot of trial and error to test different values and finding which combinations yielded the features that gave the best predictive NBR models for each city.

Once I had the optimal features for each city I ran them through the function `get_best_model` which automatically cross validated on the train and test datasets to determine the best value for the learning rate parameter for each model. After finding the best learning rate based on the lowest validation loss this model then retrains the model at the desired learning rate on all the observations from both the train and validation datasets.

I realize now that I should have built more functions like this for each stage of my work on this challenge. Functions make it so much easier to fine tune hyperparameters and find optimal features for model building but I found that process really hard and time consuming. After I had worked through a particular process using 8-10 individual code blocks in my Jupyter Notebook I realized that it would be much better to create one simple function to do all of those tasks. Building these functions was a real challenge to me, especially as I had to move between dataframes and numpy arrays for different functions and keeping the dimensions of my data in the correct format all the way through. The other real problem for me was that every time I wanted to test out a different idea or change something it would take five times longer to make the change in the function (and not be easily recovered if the idea was not a success) so I often just modified my chain of Jupyter Notebook cells. I know that this is something that I will need to improve on future projects.

Finally, I loaded the test dataset as a dataframe and used sent all the features through my `preprocess_data_lasso` function for the preprocessing, made predictions using the best NBR model generated for each city and then changed the output predictions to integers and formatted them to submission dataframe for submittal to the competition.

## Refinement

For me this entire project has been about refinement. I feel like I have a solid framework for building a predictive model for this case, but I have had little luck finding a solution I find satisfactory. I started from the Benchmark model from the DrivenData competition and the MAE error metric of 25.8173, as prescribed by this competition. This result was straightforward to replicate, and my plan was mostly to focus on feature engineering and selection to improve/refine the model and its results.

I used a loop to cross validate my models against different learning rates and build separate models for San Juan and Iquitos and this worked well from the beginning, so I have spent almost all of my time on this project searching for ways to improve the predictive power of my model.

I started by simply creating lagged features for the training data and hand selecting the best ones based on simple correlation results to include in the NBR model. This resulted in many models that produced satisfactory (to me) results based on the validation dataset (in the range of MAE 16-19) but most every time I made predictions on the test features and submitted them for analysis the results were poor, most worse than the Benchmark model, in the range of MAE 25 – 30 and sometimes as bad as 50+. But this process also produced my best scoring model with an MAE 24.4712 and earning me a rank of 496<sup>th</sup> out of over 5100 competitors.

Next, I tried PCA for feature selection to help find the features that would produce the best NBR model. I looked at the top 10 PCA components and examined the *explained\_variance\_ratio\_* of each as well as the sum of *explained\_variance\_ratio\_* for the top 10 PCA components in order to choose which PCA components should be used in the NBR model. Even though PCA only selects coefficients based on maximizing retained variance and not a metric designed to maximize the fit of a regression I hoped that it would help the NBR model in two ways; 1. Finding latent features in the training dataset with more signal to noise ratio than individuals features combined, and 2. Remove any problems resulting from collinearity of input features because each PCA component is by definition orthogonal to all the others. This did not improve the error metric and I consistently got MAE scores on the test data in the range of 25-30.

Next I tried to use *sklearn.feature\_selection.SelectKBest* to help automate the process of selecting the features that produce the best regression model for the *total\_cases* target.

Finally, in the final product I used Lasso (L1) Regularization because it helped to automate the process of choosing the best features to include in the NBR model and it also helps with problems associated with multicollinearity of input features. I built an L1 Regularization for each city and varied the alpha parameter between 0-1 for Iquitos and 0-5 for San Juan. Then I took the non-zero coefficient features and used those to create the NBR model for each city. After running many different combinations, I found the best features to use for each city by finding the minimum validation loss from the NBR model. This is what finally produced a model that made a noticeable improvement over the benchmark model created by DrivenData.

## IV. Results

---

### Model Evaluation and Validation

Due to the nature of this competition and the submission of final predictions for scoring sensitivity analysis is difficult. But I can confidently say that due to the fact that my models consistently perform worse on the test data scored by DrivenData than on my validation data that my models were not robust. Now that I know it is not just a simple overfitting issue, I believe that this challenge is just part of the nature of a time series dataset involving complicated disease transmission and environmental factors.

The parameters for all phases of my model were chosen based on cross validation of the features or model they yielded .and I feel confident in my choices. I do not feel like this model is robust enough to call it a *solution* to this forecasting challenge because it is very sensitive to small changes. I tested this model (and the its iterative predecessors) on a wide range of data inputs. I started by simply the relative size of the training and validation datasets and the results were very different. At one point I was concerned that the model was not generalizing well because older dengue total\_cases data (some of the training data is over a decade older than the test data) and no longer relevant so I trained and tested the model on datasets where I removed the oldest data, year by year to only to find out that the changes were significant and did not follow an obvious pattern. I was not until I finally visualized the training data on a year-by-year basis that I could determine that some time periods of the data were easier to model (relatively low, regular, seasonal spikes in total\_cases) and some were much more difficult (very irregular spikes or extremely large spikes). Sometimes an easier section would be in the validation data and then would give me artificially good loss scores that would generalize as well. Other times there would be a difficult part of the times series in the training or validation set but not in the other and the loss values from that section were significantly higher than expected and cause me to re-evaluate the entire model.

I do think that this model can be trusted to make predictions that fall within the achieve MAE metric of under 25 but at the moment it cannot do any better than that.

### Justification

My final model was only able to achieve an improvement of about a 1 point decrease in the competition error metric, MAE, from 25.8173 to 24.8582. Even though this final model does not feel

satisfying to me I am confident that I have exhausted every technique I could find toward this challenge. I did make some improvement to this model, but I would not call this a solution to the problem of making this forecast. The noisy nature of the total\_cases target variable combined with the environmental data as proxy for the transmission vector of mosquitos is a non-trivial challenge which I assume it was brought to a crowd-sourced challenge like DrivenData.

I would not call my model robust but it will consistently make predictions on new data with an MAE under 25 which is something I hope.

As I thought about this challenge and the real-world effects of dengue fever I doubt that this is a significant improvement at all. Even the best models in this competition have only been able to achieve MAE scores or a little more than 13 and I wonder how significant even those outstanding results will be to real world stakeholders because the average number of weekly cases of dengue fever in San Juan and Iquitos is only about 34 and 7.5, respectively. With an average error of over 13 per observation I wonder if this type of modeling is mostly useful for finding the larger trends and get some predictive insight based on whether the trend is likely to increase or decrease and how quickly, possibly similar to quantitative analysis of the stock market.

## V. Conclusion

---

### Free-Form Visualization

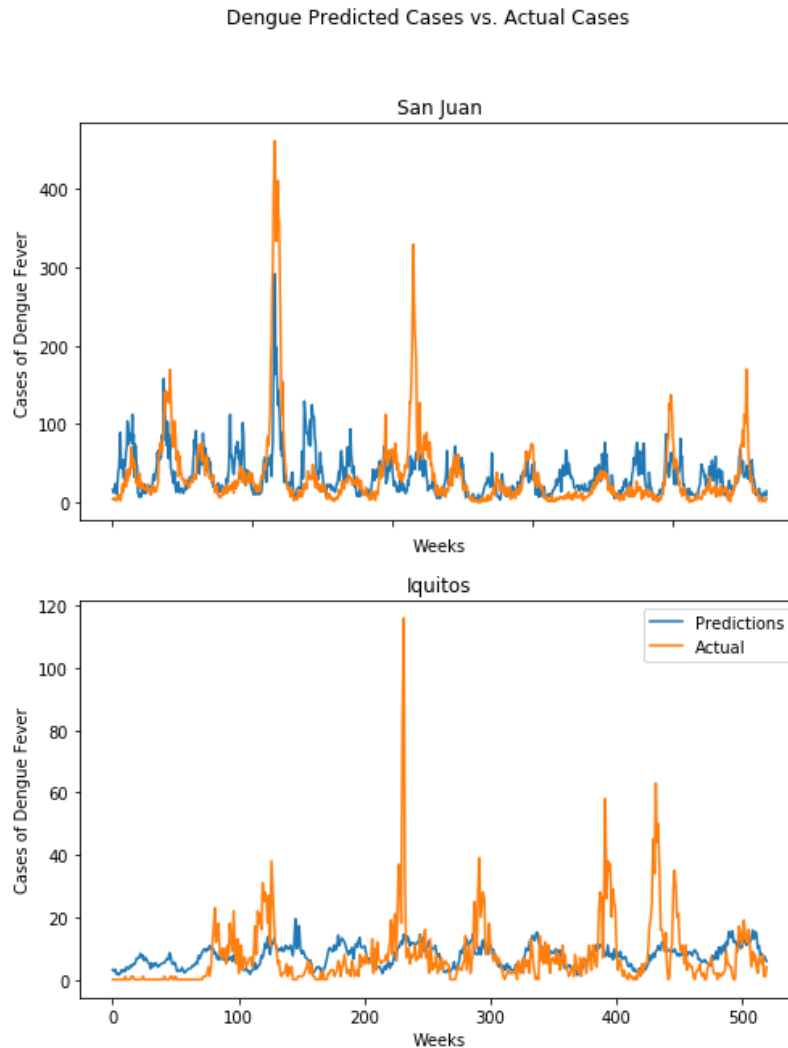


Figure 4 – NBR predictions vs actual total\_cases from training data

These two visualizations are the ones that used all through this project as a sanity check every time the NBR model produced predictions. I think these two plots show a couple of important characteristics that I kept in mind while working on this challenge. One is just how noisy and irregular this data is. The San Juan total\_cases has a fairly regular frequency, but the amplitude varies significantly. The Iquitos total\_cases is nowhere close to regular in terms of frequency or amplitude. This is something I had to constantly remind myself because it makes the models very sensitive to different sections of this data and the loss results would change drastically. These plots helped me to put those mysterious interactions into perspective. When I first started on this project



and I was not looking at plots of the data and predictions regularly then I had some significant misconceptions about the patterns in this data.

Another important point I took note of is the scale of the `total_cases` variable and how valuable this data analysis and predictions will hopefully be to someone in these cities. Iquitos has an average of around 7 dengue cases per week but has periodic spikes up around 50 and one huge spike over 100 cases per week. San Juan regularly sees around 100 cases per week but only averages about 34 but can have 300+ cases per week. These huge increases would be totally overwhelming in terms of medical staff and supplies to help this huge increase in sick people. I think that keeping the real world consequences in mind when working on a project like this helps me to better focus on insights that will be useful to the stakeholders, even though in this case I will never know them directly.

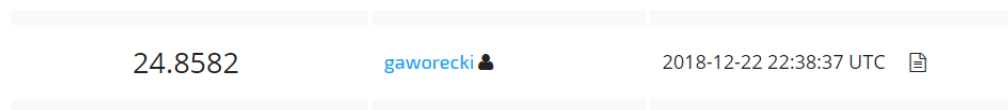
## Reflection

I started this project with very little understanding of dengue fever spread or regression modeling beyond simple linear or logistic regression. The DrivenData competition had an open source model that I used as a benchmark and I love the idea of using my new data skills to benefit others so I chose this as my capstone project. I learned all about Negative Binomial Regression and Poisson distributions as I built a Negative Binomial model to predict the number of dengue fever cases based on only environmental features. This model was relatively straightforward to research and build so I moved on to fine tuning my model to achieve the desired level of success.

The most difficult part of this project was that I found my trained model achieving respectable MAE scores of around 16 on validation data but every time I made predictions on the DrivenData test dataset and submitted them for the competition the MAE scores were around 25+, slightly worse than the benchmark model.

At first I figured that this must be the result of overfitting of my model in the training process but after much cross validation I determined this must not be the case because my validation loss was consistently lower than my training loss. So I spent many, many hours tweaking hyperparameters and trying to ensemble my NBR model with more traditional time series models like ARIMA. I even built an entire RNN Network with LSTM nodes to forecast and I learned a ton about LSTM modeling (in both Keras and Pytorch) but I could not get the model to make any better predictions on the DrivenData test dataset for the competition.

I have spent an immense amount of time research and testing every technique I could find for improving my NBR model but unfortunately, I was only able to achieve an improvement of dropping the MAE error metric by about 1 point. This is much less of an improvement than I originally planned on but I did learn that this type of quasi-seasonal data is very difficult to model. I finally discovered (after many weeks of diving into the data and processing and modeling it in all the ways described above) that the models are very sensitive to which part of the time series is used in the training data vs the validation data vs the test data. Every few years there are spikes in the data that are an order of magnitude higher than even a typical rolling 30 day high and depending on how many of these spikes exist in these different datasets it can cause a model to train poorly or just not generalize to other data well at all. I am very excited to work on a problem like this one in the future and draw from this experience and pursue many more techniques to develop a more robust model.



A screenshot of a table showing top model submission scores. The table has three columns: score, username, and timestamp. The score is 24.8582, the username is gaworecki, and the timestamp is 2018-12-22 22:38:37 UTC. There is a small icon next to the username.

|         |           |                         |
|---------|-----------|-------------------------|
| 24.8582 | gaworecki | 2018-12-22 22:38:37 UTC |
|---------|-----------|-------------------------|

Figure 5 – screenshot of top model submission scores from DrivenData

## Improvement

I do not think that there is much more improvement to be gained using a Negative Binomial Regression model. That said I had planned to try ElasticNet regularization to see if that would give my model the boost I have been searching for but I have unfortunately run out of time to try this alternative. I would also like to investigate more advanced feature engineering techniques to see if there are gains to be found there as well but I would need to find a mentor for this challenge.

The improvement that I would like to make would be to use a multivariate time series model that could use the environmental conditions along with the total\_cases of dengue fever at previous time steps to make better predictions. I would particularly like to use an LSTM model because it seems like the perfect type of model for this challenge. I have actually built an LSTM model but after many hours I could not get it to make better predictions than my NBR model. Most of the example models I have studied were simple univariate time series models that were not applicable to this problem which made using an LSTM model difficult. I used the few multivariate LSTM examples I found to

build my model, but I had similar problems to my other model in that it achieves respectable loss results but when used to make predictions on the test dataset the loss is significantly worse.

At this point I am eagerly waiting for this competition to end so that I can investigate the code for the top performing models and see how they did it. I suspect, and LSTM model will be critical to at least some of their successes.