Capstone Project
Machine Learning Engineer Nanodegree

Giulia Martielli
March 20th, 2021

Predicting Dengue Disease Spread

Figure 1 - Image from unsplash.com

Definition

Domain Background

In 2019 I had the chance of visiting an astonishing country, Sri Lanka. During our stay there, although we were already aware of it and had taken the necessary precautions, we had a few conversations with local people about the phenomenon of Dengue and the risk that one can run every day by being bitten by a particular species of mosquitoes, *Aedes* mosquitos, that carry this disease. From there my curiosity about this phenomenon grew and I began to wonder if it was possible in some way to use statistical and / or machine learning techniques to analyse and predict it. This project of mine, therefore, is based on a competition held by the site DrivenData.org, called "DengAI: Predicting Disease Spread", which focuses on finding a way of predicting the next dengue fever local epidemic in *San Juan*, Puerto Rico and *Iquitos*, Perú.

Problem Statement

As stated on the DrivenData.org competition page, in the past years dengue fever (DF) has been most prevalent in Southeast Asia and the Pacific Islands. Nowadays, however, it is much more spread around the world and many of the nearly half billion cases per year are occurring in Latin America. According to Salles *et al.* (2018), the virus has become a major threat to American human life, reaching approximately 23 million cases from 1980 to 2017.

DF is a systemic and dynamic infection with a broad clinical spectrum that includes both serious and non-serious clinical manifestations. In mild cases, symptoms are similar to the flu: fever, rash, and muscle and joint pain. In severe cases, dengue fever can cause severe bleeding, low blood pressure, and even death (WHO,

2009). No effective therapy for dengue exists at the moment; treatment is purely symptomatic, requiring a high level of patient care. This is why predicting upcoming epidemics can be a valuable aid.

Since DF is carried by mosquitos, its transmission is higly correlated with climate variables, especially temperature and precipitation / humidity. The task of the competition is to *predict the number of dengue cases each week* (in each of the two locations – San Juan and Iquitos) based on all the environmental variables provided (temperature, precipitation, vegetation, and more).

Datasets and Inputs

The data for this project are downloaded directly from DrivenData.org (more information about the sources of the data can be found at this link). In addition to the number of cases in the two locations, the data include, as previously mentioned, information on temperature, precipitation, humidity, vegetation, and what time of the year the data was obtained. Below a detailed list and description of datasets provided and features.

Datasets:

- `dengue_labels_train.csv` (1,457 rows) containing four columns being `city` (`sj` for San Juan and `iq` for Iquitos), `year` (ranging from 1990 to 2008 for San Juan and from 2000 to 2010 for Iquitos), `weekofyear` (ranging from 1 to 53) and `total_cases` (the number of cases/week for each city);
- `dengue_features_train.csv` (1,457 rows) containing data for both San Juan (years ranging from 1990 to 2008) and Iquitos (years ranging from 2000 to 2010), plus all the relevant features, for a total of 24 columns;
- `dengue_features_test.csv` (417 rows) containing data for both San Juan (years ranging from 2008 to 2013) and Iquitos (years ranging from 2010 to 2013), plus all the relevant features, for a total of 24 columns; as reported on the DrivenData.org website, "the test set is a pure future *hold-out*, meaning the test data are sequential and non-overlapping with any of the training data".

Here is a list of all the features contained in the train and test dataset:

- City and date indicators
  - `city` – City abbreviations: `sj` for San Juan and `iq` for Iquitos
  - `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

- City and date indicators
  - `city` – City abbreviations: `sj` for San Juan and `iq` for Iquitos
  - `week_start_date` – Date given in yyyy-mm-dd format

- 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 (0.5x0.5 degree scale) measurements
  - `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

Sample values for each feature can be found below.

Features	Sample Values
city	sj
year	1990
weekofyear	18
week_start_date	30/04/1990
ndvi_ne	0.1226
ndvi_nw	0.103725
ndvi_se	0.1984833
ndvi_sw	0.1776167
precipitation_amt_mm	12.42
reanalysis_air_temp_k	297.572857143
reanalysis_avg_temp_k	297.742857143
reanalysis_dew_point_temp_k	292.414285714
reanalysis_max_air_temp_k	299.8
reanalysis_min_air_temp_k	295.9
reanalysis_precip_amt_kg_per_m2	32.0
reanalysis_relative_humidity_percent	733.657142857
reanalysis_sat_precip_amt_mm	12.42
reanalysis_specific_humidity_g_per_kg	140.128571429
reanalysis_tdtr_k	262.857142857
station_avg_temp_c	254.428571429
station_diur_temp_rng_c	6.9
station_max_temp_c	29.4
station_min_temp_c	20.0
station_precip_mm	16.0

Table 1 - Sample values for every feature in the datasets

The goal is to predict the `total_cases` label for each triplet (`city`, `year`, `weekofyear`) in the test set. Another important consideration to report is that throughout the datasets, missing values have been filled as `NaNs`. After performing some exploratory analysis to the data it will be evaluated whether the missing data might need to be filled / replaced through some sort of data imputation (mean value for continuous features – for instance) and whether there are features which might be removed from the dataset (if not significant or redundant), in order to simplify the analysis.

Evaluation Metrics

The evaluation metric suggested for the DrivenData.org competition is the *mean absolute error (**MAE**)*, which is a model evaluation metric usually used with regression models. "The mean absolute error of a model with respect to a test set is the mean of the absolute values of the individual prediction errors on over all instances in the test set (n). Each prediction error is the difference between the true value (y_i) and the predicted value for the instance ($\hat{y}_i$)" (C. & G.I., 2011).

$$MAE = \frac{\sum_{i=1}^{n} abs(y_i - \hat{y}_i)}{n}$$

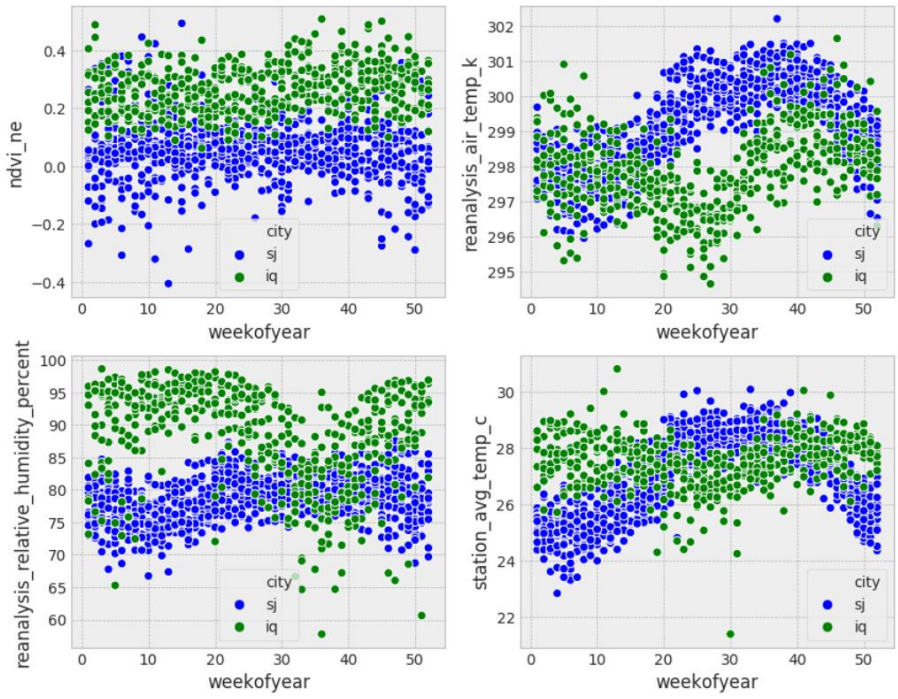
Other graphic evaluation criteria are considered, such as the actual vs predicted values plot.

Analysis

Exploratory Data Analysis

Are there differences between the two cities?

After performing a first exploratory analysis on basic statistics for both train and test set, a thing to be considered in such a dataset is whether the two cities in considerations San Juan (Puerto Rico) and Iquitos (Perú) contain substantial differences in their distribution of variables representing vegetation, temperature, and humidity. Below four plots showing the different behaviours by city, of four of the most significant variables in the train set.

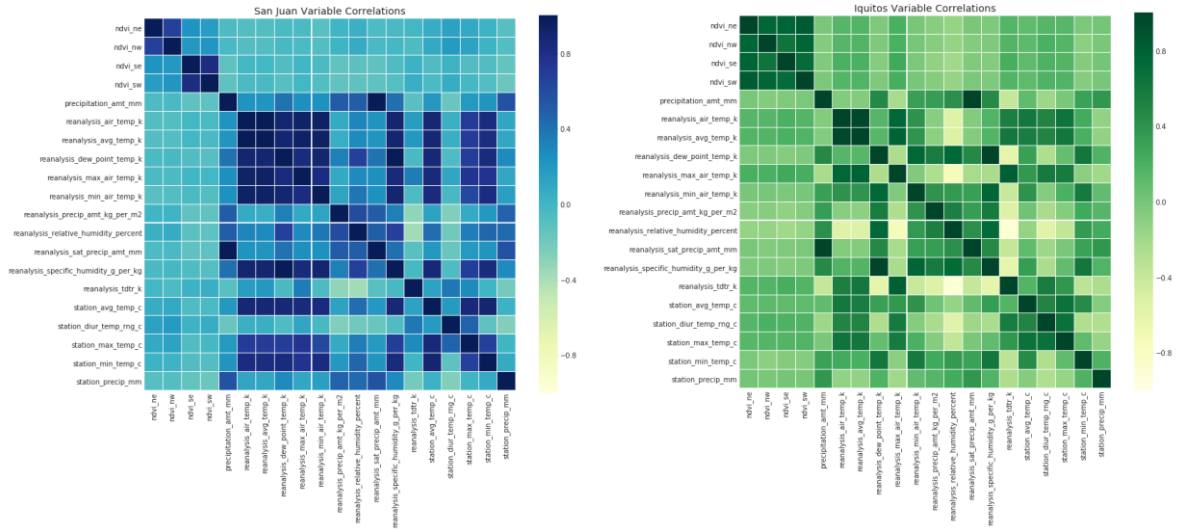


As one can see, the distributions for the two cities seems very different from each other. Probably because San Juan belongs to the Northern hemisphere and Iquitos belongs to the Southern hemisphere. In fact, all the variables related to the distribution of vegetation, and those related to humidity, temperature and precipitation seem to have almost an opposite behaviour in the two cities with respect to the weeks of the year. For example, in San Juan average temperatures are highest around week 30 of the year (full summer in the Northern hemisphere), as well as humidity is lowest in this period; while on the contrary in Iquitos, the

average temperature is highest around weeks 52-1 (corresponding to summer in the Southern hemisphere), as well as humidity is lowest in the same period. For this reason, ***the two cities will be treated separately*** from now on and different models will be fitted on them.

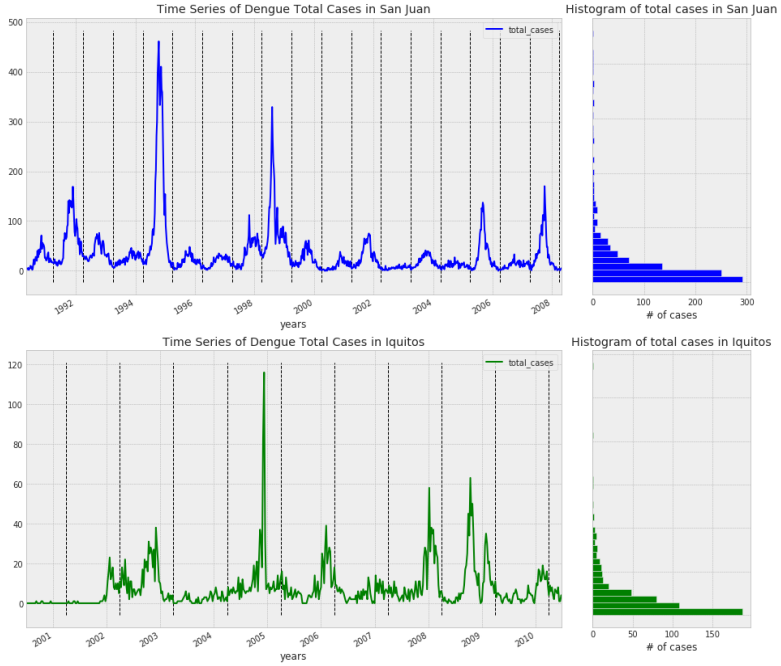
Features Correlations

The two plots below show *highly correlated* features in the train sets for both San Juan and Iquitos. In the Data Pre-processing Section these features will be removed and then the same will be done in the test set, to avoid any data leakage problems and in order to lower model complexity. This difference in the correlations of these cities supports the choice to keep the two cities separate.



Target Variable Distribution and Correlations

The following plots show the combination of the historical series of the total number of cases in the two cities in the various years considered, along with their relative frequencies.



As one can see, San Juan had several peaks of cases in 1995, 1999 and then two more moderate peaks in 2006 and 2008, respectively; while Iquitos had its highest peak of infections in 2005 - a year when cases in San Juan were very low - followed by 2009 and 2008.

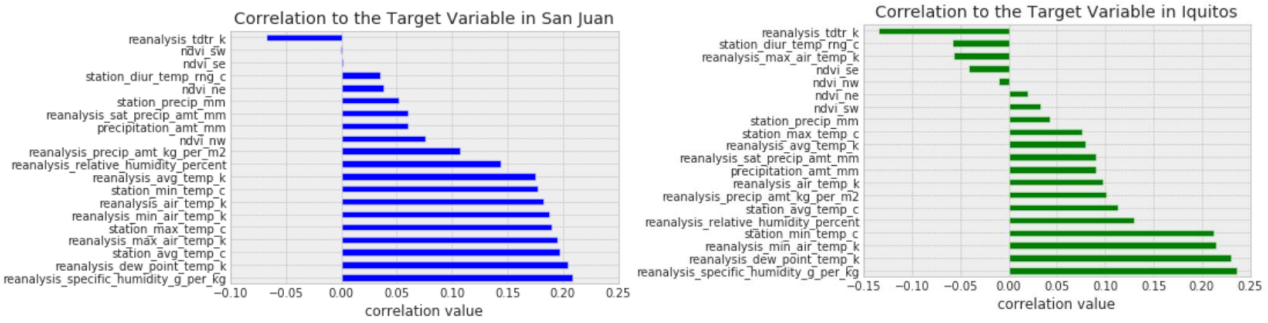
The table below summarises the basic statistics for the target variable `total_cases` in the train sets.

Statistics	San Juan	Iquitos
count	933	518
mean	34.18	7.58
std	51.45	10.78
min	0.00	0.00
25%	9.00	1.00
50%	19.00	5.00
75%	37.00	9.00
max	461	116

Table 2 - Basic statistics for the total number of dengue cases in San Juan and Iquitos

Once again, the differences between the two cities are confirmed. The average number of cases per week in San Juan are more than four times those in Iquitos; the first city also has a much higher variance, probably this is also due to the difference in population density between the cities. In fact, even though they have similar population numbers, the density in San Juan is about 1.980 people/km², while in Iquitos it is just 326.86 people/km² (Wikipedia, 2021). The minimum and maximum values of the target variable are also very different.

The two bar-plots below show the correlation between each feature and the target variable total cases of dengue.



The target variable `total_cases` seems not to be correlated with the two features related to vegetation `ndvi_sw` and `ndvi_se` in the city of San Juan, for this reason in the pre-processing phase these variables will be removed from both train and test set, to further simplify the model. While in the city of Iquitos the target variable seems to be correlated - albeit minimally - to all features.

Algorithms and Techniques

The task of predicting the total cases of dengue for the next *X* weeks in the future is a *supervised* (time series, more specifically) problem, since we know that the target variable is a numeric discrete variable (number of cases). For this reason, every forecasting algorithm / model able to support features with different natures as an input might be appropriate. Some examples range from classic statistical forecasting methods (Simple Exponential Smoothing, ARIMA etc.), to machine learning methods, such as Gradient Boosting Trees (XGBoost), and to Recurrent Neural Network (such as Long Short-Term Memory – LSTM).

Since it might be interesting to know how classical statistical models perform compared to machine learning models and neural networks, in this project two different supervised learning models are taking into consideration and applied to the data, to check whether at least one of them performs better than the classic statistical benchmark model (more details in the next sections).

Extreme Gradient Boosting (XGBoost)

XGBoost stands for *Extreme Gradient Boosting* and it is an efficient implementation of the stochastic gradient boosting. It is both fast and efficient and it performs well on a wide range of predictive modelling tasks; for this reason, it is one of the most popular algorithms used in data science competitions: "tree boosting has been shown to give state-of-the-art results on many standard classification benchmarks" (Tianqi C. and C. Guestrin, 2016). According to (Brownlee J., 2020), from a technical point of view, it represents an ensemble of decision trees algorithms where new trees fix errors of those trees that are already part of the model. Trees are added until no further improvements can be made to the model. XGBoost is designed mainly for classification and regression tasks, but thanks to its scalability in all scenarios, it can also be used for time series forecasting problems.

Long Short-Term Memory (LSTM)

A Recurrent Neural Network (RNN), unlike a traditional neural network, can "remember" what happens in the past and takes this into account when formulating the output. However, they have a con: they cannot handle "long-time dependencies" well. This means that they remember past events, but they are not able to store events too far back in time. *Long Short-Term Memory* networks - usually simply called "LSTM" - are a special type of RNN that can handle long-term dependencies (Brownlee J., 2020). For this reason, they are among the most widely used types of neural networks to date.

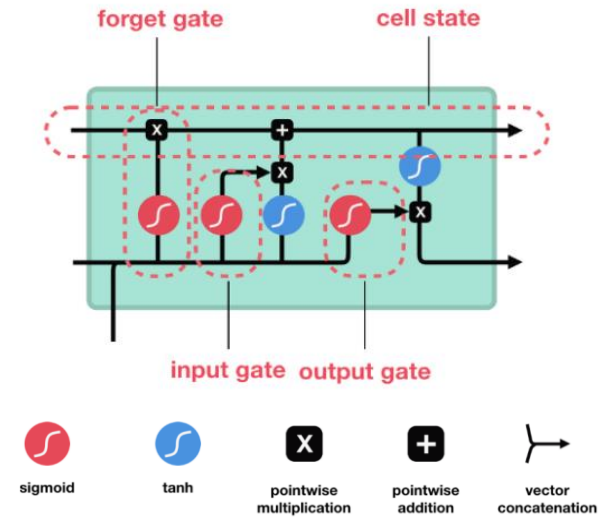


Figure 2 - LSTM architecture (Phi M., 2020)

In order to overcome the RNN vanishing gradient problem, happening when the gradient shrinks as it back-propagates through time, Hochreiter S. and Schmidhuber J. (1997) added to a classic RNN architecture the so-called *gates*, which regulates the flow of information (as shown in the figure on the left). They decide which information is important to keep and which can be thrown away.

According to Hochreiter S. and Schmidhuber J. (1997) "truncating the gradient where this does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete-time steps by enforcing constant error flow through constant error carousels within special units".

Benchmark Model

As a benchmark model, the one reported onto the DrivenData.org website itself at this link will be considered. The model is a *Negative Binomial regression*, which is usually indicated when the data distribution suggests a much larger variance than the mean. The benchmark model has its flaws and has space for further improvements, as also stated on the website, since by taking a look at its actual vs predicted plot 1) the timing of the seasonality of the predictions has a mismatch with the actual results and 2) the predictions are relatively (too) consistent, they miss the spikes that represent the large outbreaks (hence the most problematic periods). This model, though, will represent a great starting point for comparison.

The features used by the benchmark model are:

- `reanalysis_specific_humidity_g_per_kg`

- `reanalysis_dew_point_temp_k`
- `station_avg_temp_c`
- `station_min_temp_c`

Results will be presented in the Benchmark Model Section.

Implementation

Data Pre-processing

Missing Values

During the exploratory data analysis performed on the data, missing values are reported for both cities' datasets. All missing values at this stage are imputed with the mean of the features in datasets themselves, since – as Kuhn, M. and Johnson, K. (2019) state, they are common occurrences in data, though most predictive modelling techniques cannot handle them. Therefore, this problem must be addressed prior to modelling. Mean values are calculated on the train set only and those values are used to perform imputation for the test set too, to avoid any data leakage issue that may arise.

Outliers

Along with missing values, during the exploratory analysis, also outliers are checked and are handled with *flooring* and *capping* techniques, based on the tenth and ninetieth percentiles, respectively, of the distribution of the features in the trainsets of the two cities. Those values are used to cap and floor also the test sets.

Features Selection

Feature selection is done based on exploratory data analysis previously performed on the data. First feature with a high level of pairwise correlation (≥ 0.90) are removed, keeping only one among them. Again, this analysis is performed on the train sets for the two cities and then reported to the test sets. Also, columns that have resulted not to be correlated to the target variable, as seen in the Exploratory Data Analysis Section, for the San Juan data, (`ndvi_sw` and `ndvi_se`) are removed from the train and test set.

The resulting features are shown in the table below.

Features	San Juan	Iquitos
city	X	X
year	X	X
weekofyear	X	X
week_start_date	X	X
ndvi_ne	X	X
ndvi_nw	X	X
ndvi_se		X
ndvi_sw		X
precipitation_amt_mm		
reanalysis_air_temp_k		X
reanalysis_avg_temp_k		
reanalysis_dew_point_temp_k		
reanalysis_max_air_temp_k		
reanalysis_min_air_temp_k	X	
reanalysis_precip_amt_kg_per_m2		
reanalysis_relative_humidity_percent		
reanalysis_sat_precip_amt_mm	X	X
reanalysis_specific_humidity_g_per_kg	X	X
reanalysis_tdtr_k	X	X

station_avg_temp_c	X	X
station_diur_temp_rng_c	X	X
station_max_temp_c	X	X
station_min_temp_c	X	X
station_precip_mm	X	X

Table 3 - Features selected are indicated with an 'X'

XGBoost Implementation

First model to be implemented is XGBoost. For both cities the respective train sets are split into two data subsets, 80% of the train data are kept as training set and the rest is used as validation set. Function `XGBRegressor`, from `xgboost` package is used to fit the model to the data.

San Juan

The first model is implemented with no added features, by only fine tuning the hyperparameters.

```
xgb.XGBRegressor(n_estimators = 1000,
                                 learning_rate = 0.001,
                                 max_depth = 10,
                                 subsample = 0.8
                                 colsample_bytree = 0.75,
                                 objective ="reg:linear")
```

Resulting MAE is 17.84, actual vs predicted plot can be found in the XGBoost Implementation Appendix Section, Figure 3. It looks like the predictions are a little bit late in detecting peaks in total cases. To check for improvements, *moving average rolling features* are added to the model. After performing some tests, rolling means at 7 and 14 weeks are added to every chosen feature for the city. Results improve, showing a MAE of 11.49, actual vs predicted plot can be found in the XGBoost Implementation Appendix Section, Figure 4. Plot showing actual vs predicted total cases for all different models fit to the data is shown in the Model Evaluation and Justification Section.

Final predictions for the test set are made and can be found into the *predictions* folder in the GitHub repository; an extract of such predictions can be found in the Predictions Section.

Iquitos

The first model is implemented with no added features, by only fine tuning the hyperparameters.

```
xgb.XGBRegressor(n_estimators = 150,
                                 learning_rate = 0.01,
                                 subsample = 0.8,
                                 colsample_bytree = 0.75)
```

Resulting MAE is 7.06, actual vs predicted plot can be found in the XGBoost Implementation Appendix Section, Figure 5. In this case, the model is clearly too *timid*, meaning that it does not catch the peaks of total dengue cases at all and at the same time it seems like it is *overfitting*. *Moving average rolling features* are added to the model to check for any improvement. After performing some tests, also on the hyper parameters, rolling means at 1, 5, 10 and 15 weeks are added to every chosen feature for the city. Hyperparameters for the final model are given below.

```
xgb.XGBRegressor(n_estimators = 10,
                                 learning_rate = 0.15,
                                 subsample = 0.8,
                                 colsample_bytree = 0.75)
```

Results improve though not much, showing a MAE of 6.37, actual vs predicted plot can be found in the XGBoost Implementation Appendix Section, Figure 6. Plot showing actual vs predicted total cases for all different models fit to the data is shown in Model Evaluation and Justification Section.
Final predictions for the test set are made and can be found into the *predictions* folder in the GitHub repository; an extract of such predictions can be found in the Predictions Section.

LSTM Implementation

Second model to be implemented is a Bidirectional (in order to better capture time series trend and preserve information from both past and future) LSTM recurrent neural network. For both cities the respective train sets are split into two data subsets, 80% of the train data are kept as training set and the rest is used as validation set. Package used for the RNN is `keras` (Using `TensorFlow` backend); it is used for fitting the model itself and for evaluating it.

San Juan

After several trials, the best results are achieved with the following neural network structure of and the following (hyper)parameters, using all the 11 features selected from the feature selection process and also creating 50 time-steps (lagged features) for every one of them.

```
epochs = 250
n_hidden = 25
activation = 'linear'

lstm1 = Sequential()
lstm1.add(Bidirectional(LSTM(n_hidden, input_shape=((lag,feat)))))
lstm1.add(Dropout(0.4))
lstm1.add(Dense(n_hidden))
lstm1.add(Dropout(0.4))
lstm1.add(Dense(1))
lstm1.compile(loss='mae', optimizer='adam', metrics =
['mean_absolute_error'])
model_lstm1 = lstm1.fit(df_train_sj_x,df_train_sj_y,validation_data =
(df_val_sj_x,df_val_sj_y), epochs=epochs, batch_size=72, shuffle=False)
```

Resulting MAE in the train set is 15.40, while MAE in the validation set is 16.52 – indicating a slight *overfit* to the data; model loss plot and actual vs predicted plot can be found in the LSTM Implementation Appendix Section, Figure 11 and Figure 12. The neural network for San Juan data needs quite a few epochs before settling and still the losses from both train and validation sets result to be quite oscillating and unstable. Plot showing actual vs predicted total cases for all different models fit to the data is shown in the Model Evaluation and Justification Section.
Final predictions for the test set are made and can be found into the *predictions* folder in the GitHub repository; an extract of such predictions can be found in the Predictions Section.

Iquitos

After trying with an 80-20 rate for Iquitos case, the percentage split between train and validation is set to 60-40. This is done because the neural network has not been able to generalise well onto the validation set data well with an 80-20 split rate, due to the small number of samples. After several trials, the best results are achieved with the following neural network structure of and the following (hyper)parameters, using all the 13 features selected from the feature selection process and also creating 1 time-step (lagged features) for every one of them.

```
epochs = 50
n_hidden = 40
activation = 'linear'

lstm1 = Sequential()
lstm1.add(Bidirectional(LSTM(n_hidden, input_shape=((lag,feat)))))
lstm1.add(Dropout(0.4))
lstm1.add(Dense(n_hidden))
lstm1.add(Dropout(0.4))
lstm1.add(Dense(1))
lstm1.compile(loss='mae', optimizer='adam', metrics =
['mean_absolute_error'])
model_lstm1 = lstm1.fit(df_train_sj_x,df_train_sj_y,validation_data =
(df_val_sj_x,df_val_sj_y), epochs=epochs, batch_size=72, shuffle=False)
```

Resulting MAE in the train set is 6.55, while MAE in the validation set is 6.58; model loss plot and actual vs predicted plot can be found in the LSTM Implementation Appendix Section, Figure 13 and Figure 14. Although the value of the MAE seems to be optimal and there seems to be no instability or bias / overfitting from the plot of the loss, looking at the plot of the actual vs predicted values it is clear that the neural network is not able to capture the trend of the data in the city of Iquitos at all. In fact, the predictions are flat, very similar to a moving average. Plot showing actual vs predicted total cases for all different models fit to the data is shown in Model Evaluation and Justification Section.

Final predictions for the test set are made and can be found into the *predictions* folder in the GitHub repository; an extract of such predictions can be found in the Predictions Section.

Benchmark Comparison

Finally, the benchmark model reported onto the DrivenData.org website itself is considered. The model is a *Negative Binomial regression*, and to stick to the true version, all the steps performed on the DrivenData website are performed too, starting from data pre-processing up to the modelling itself. Formula for the model is given below and it is common for both cities.

```
model_formula = "total_cases ~ 1 + " \
                    "reanalysis_specific_humidity_g_per_kg + " \
                    "reanalysis_dew_point_temp_k + " \
                    "station_min_temp_c + " \
                    "station_avg_temp_c"
```

San Juan

Resulting MAE is **20.64**, actual vs predicted plot can be found in the Benchmark Model Implementation Appendix Section, Figure 15. The predictions in the actual vs predicted plot show a very cyclical behaviour of the total predicted cases of dengue; the model is in fact not able to capture the peaks in the city of San Juan. Plot showing actual vs predicted total cases for all different models fit to the data is shown in Model Evaluation and Justification Section.

Final predictions for the test set are made and can be found into the *predictions* folder in the GitHub repository; an extract of such predictions can be found in the Predictions Section.

Iquitos

Resulting MAE is **6.53**, actual vs predicted plot can be found in the Benchmark Model Implementation Appendix Section, Figure 16. The model seems to pick up on trends better than the benchmark model fitted for the city of San Juan, but it is still a long way from predicting peaks with any confidence. Plot showing actual vs predicted total cases for all different models fit to the data is shown in Model Evaluation and Justification Section.

Final predictions for the test set are made and can be found into the *predictions* folder in the GitHub repository; an extract of such predictions can be found in the Predictions Section.

Results

Model Evaluation and Justification

Results for models fit to data from San Juan and Iquitos cities are summarised in the table below. As for the XGBoost model, only the better of the two - i.e., the one containing the moving average rolling features - is considered.

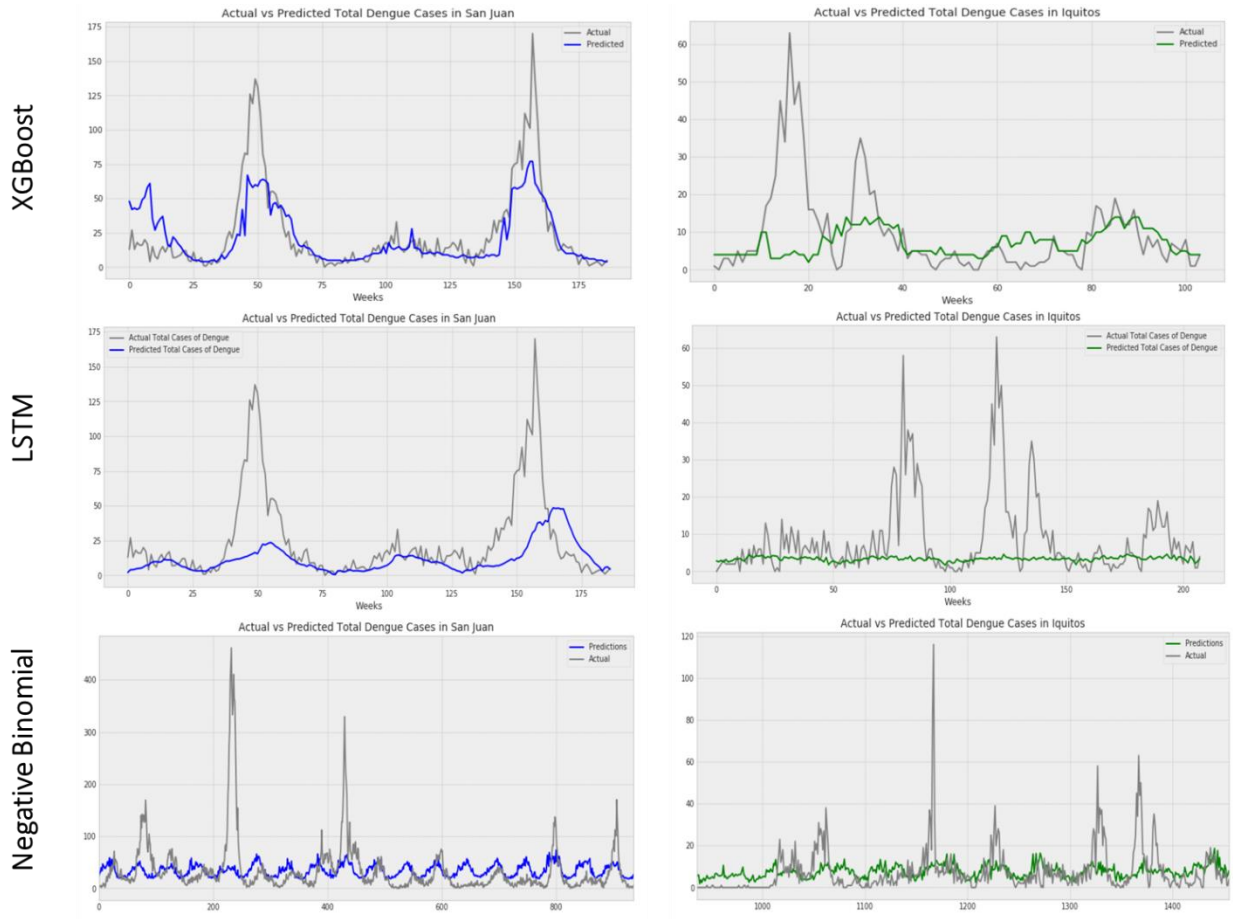
City	Model	Package	MSE (train set)
San Juan	XGBoost	SKLearn	**11.49**
	LSTM	Keras	15.40
	Negative Binomial	statsmodels	20.64
Iquitos	XGBoost	SKLearn	**6.37**
	LSTM	Keras	6.55
	Negative Binomial	statsmodels	6.53

By looking at the MSE only, the best model seems to be the XGBoost for both cities. It must be said, however, that the biggest difference can be seen in the city of San Juan where the XGBoost visibly does a better job at predicting total future cases of dengue. In the city of Iquitos, however, the difference is very small and the three models, considering only the MAE, seem to perform quite equally.

For this reason, is important to also take into consideration actual vs predicted plots for all the different models. They are shown in the next page. From the plots, the accuracy of XGBoost in predicting the phenomenon in both cities becomes much more visible, compared to the other two models.

LSTM shows poor performances in predicting the phenomenon. This might be due to the particularly small amount of training examples. It is known that in such cases, alternatives to deep learning methods are superior in performance since they do not overfit and they are often computationally superior to begin with.

Regarding the benchmark model, as stated on the DrivenData website, the model has some issues: the timing of the seasonality of the predictions has a mismatch with the actual results and the predictions are relatively (too) consistent, they miss the spikes that represent the large outbreaks (hence the most problematic periods). LSTM is better at forecasting San Juan data with respect to the benchmark, while benchmark seems to be better than the LSTM for Iquitos data.



Further Improvements

Selecting optimal parameters for a neural network architecture and creating / selecting the right features, which can help the network to learn the patterns in the data, can often make the difference between ordinary and state-of-the-art performance (Reimers N. and Gurevych I., 2017). Thus, a more rigorous approach for hyperparameters selections can be used to better tune the customized LSTM-based architecture and more specific features can be created for better representing the phenomenon. Another improvement that can be done is using ensemble techniques, for instance by mixing three based models and Neural Networks.

Conclusions

In this project, two different statistical and machine learning methods have been considered for predicting the total cases of dengue in two different cities, San Juan (Puerto Rico) and Iquitos (Perú), given their historical data, and they have been compared to a benchmark model. During the analysis, data ingestion and Exploratory Data Analysis (EDA) has been performed, to clearly understand the data before performing any sort of processing. Data have then been pre-processed and prepared for the models. The models considered are XGBoost, LSTM and Negative Binomial regression as a benchmark model. Among the three, the best in terms of performances (MAE and actual vs predicted values) has been found to be the XGBoost, with moving average rolling features added. Hence, with respect to the benchmark model, seasonality timing mismatch issue has been addressed, along with the issue of not predicting spikes at all.

References

Administration, N. O. a. A., n.d. *Dengue Forecasting.* [Online]
Available at: http://dengueforecasting.noaa.gov/

Anon., n.d. *Dengue.* [Online]
Available at: https://www.cdc.gov/Dengue/

Brownlee, J., 2020. *A Gentle Introduction to Long Short-Term Memory Networks by the Experts,* s.l.: Machine Learning Mastery.

Brownlee, J., 2020. *How to Use XGBoost for Time Series Forecasting,* s.l.: Machine Learning Mastery.

C., S. & G.I., W., 2011. *Encyclopedia of Machine Learning.,* Boston, MA: s.n.

DrivenData, n.d. *DrivenData.* [Online]
Available at: https://www.drivendata.org/competitions/44/dengai-predicting-disease-spread/

Guestrin, C. & Chen, T., 2016. *XGBoost: A Scalable Tree Boosting System,* s.l.: ArXiv.Org.

Hochreiter, S. & Schmidhuber, J., 1997. *Long Short-Term Memory,* s.l.: Neural Computation.

Kuhn, M. & Johnson, K., 2019. *Feature Engineering and Selection: A Practical Approach for Predictive Models (Chapman & Hall/CRC Data Science Series).* 1 ed. s.l.:Chapman and Hall/CRC..

M., P., 2020. *Illustrated Guide to LSTM's and GRU's: A step by step explanation,* s.l.: Medium.

Organization, W. H., 2009. *Dengue: guidelines for diagnosis, treatment, prevention and control.* [Online]
Available at: https://www.who.int/tdr/publications/documents/dengue-diagnosis.pdf

Reimers, N. & Gurevych, I., 2017. *Optimal hyperparameters for deep lstm-networks for sequence labeling tasks,* s.l.: s.n.

Syed, A., n.d. *Unsplash.com.* [Online]
Available at: https://unsplash.com/photos/nZgpg4xYhjM

T.S., S., T., D. E. S.-G. & E.S.L., D. A., 2018. History, epidemiology and diagnostics of dengue in the American and Brazilian contexts: a review.. *Parasites Vectors,* Issue 11, p. 264.

Wikipedia, 2021. *Wikipedia.* [Online]
Available at: https://it.wikipedia.org/wiki
[Accessed 03 2021].

Appendix

XGBoost Implementation Appendix

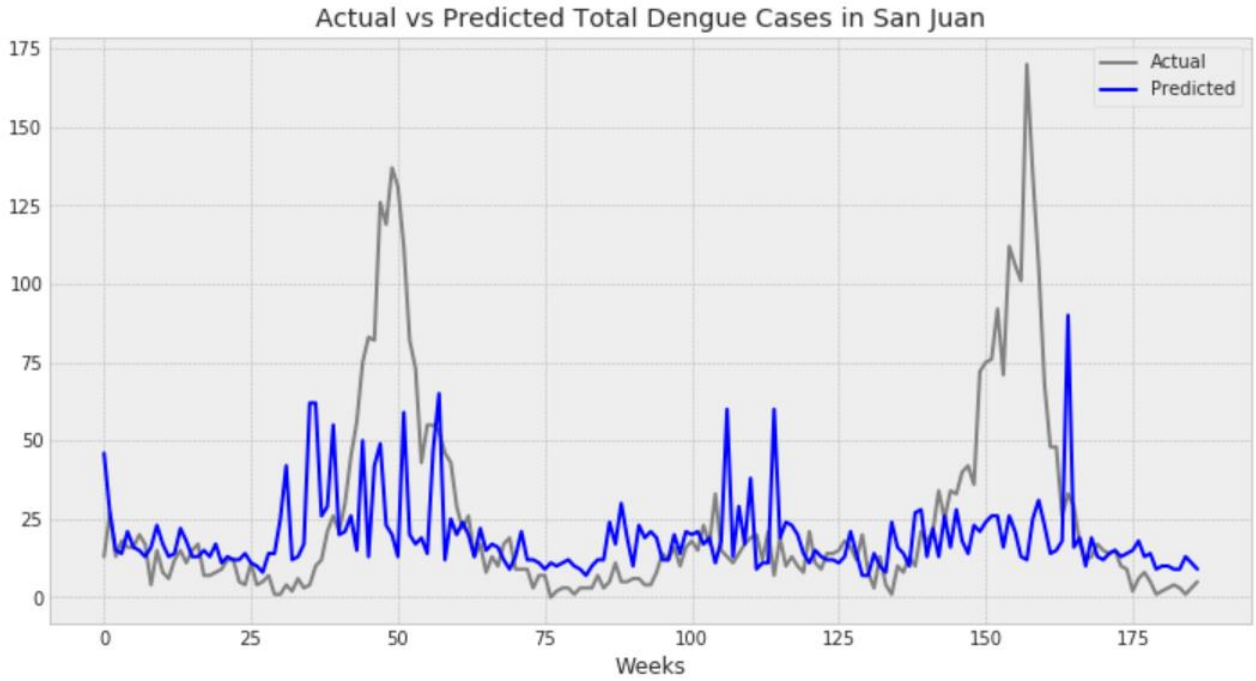


Figure 3 - Actual vs predicted total dengue cases in San Juan, XGBoost model with no added features

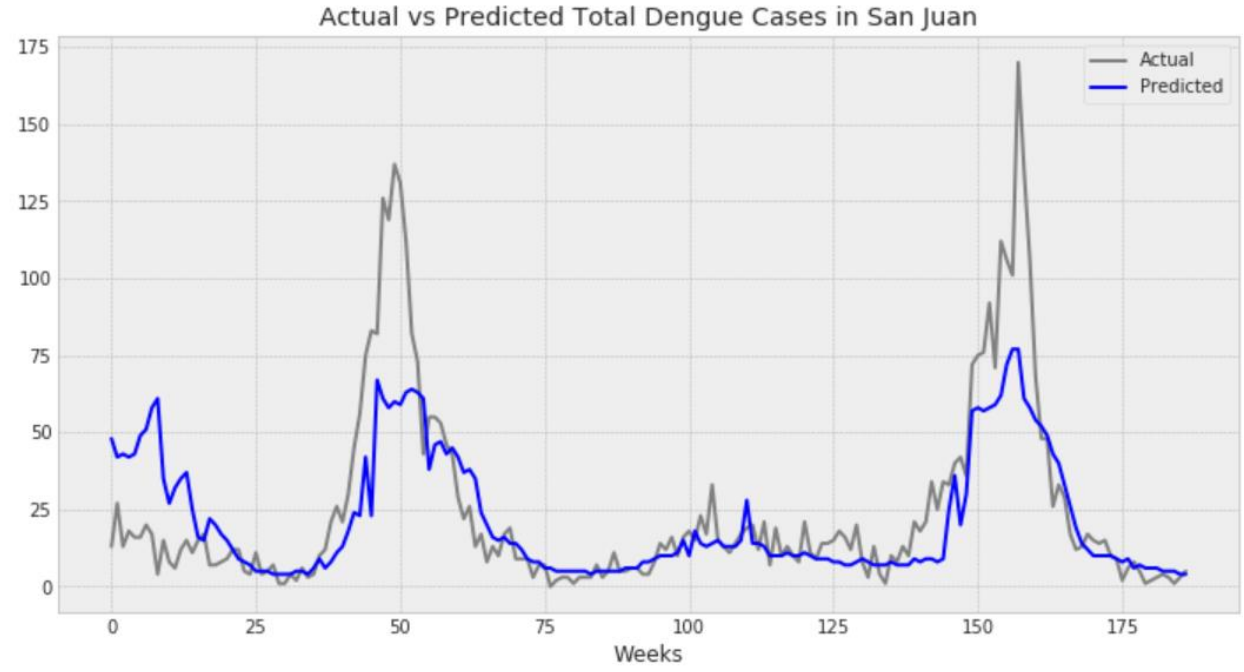


Figure 4 - Actual vs predicted total dengue cases in San Juan, XGBoost with moving average rolling features added to the model

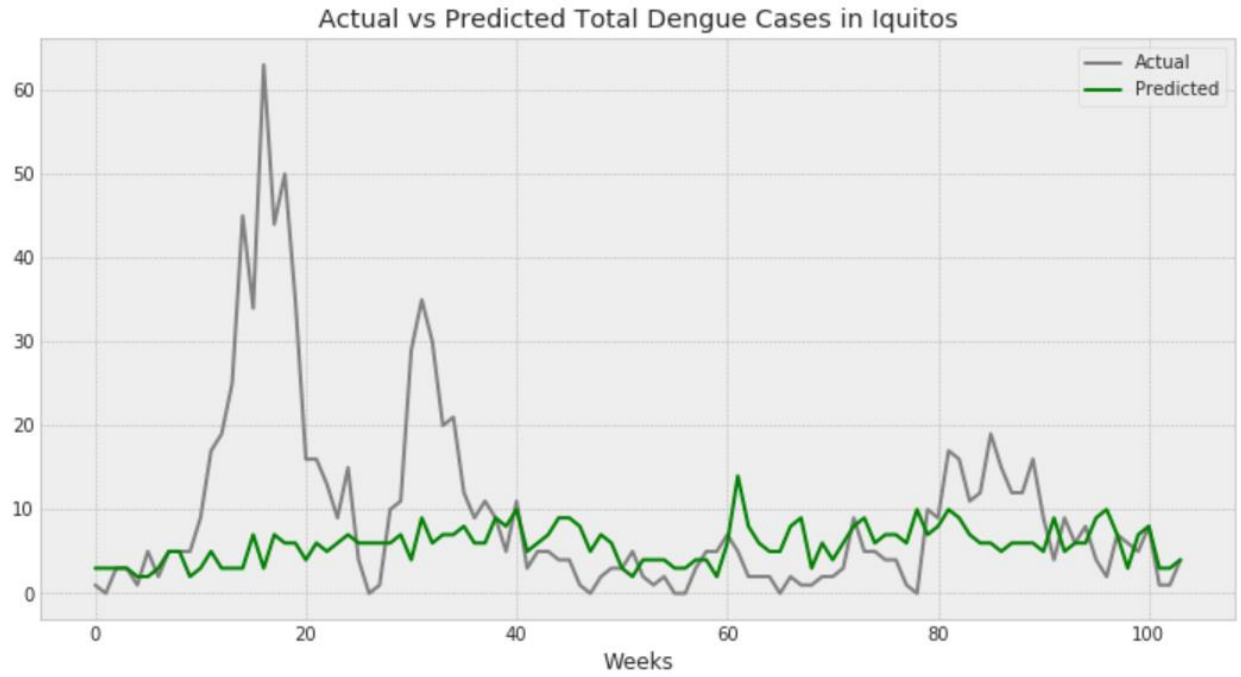


Figure 5 - Actual vs predicted total dengue cases in Iquitos, XGBoost model with no added features

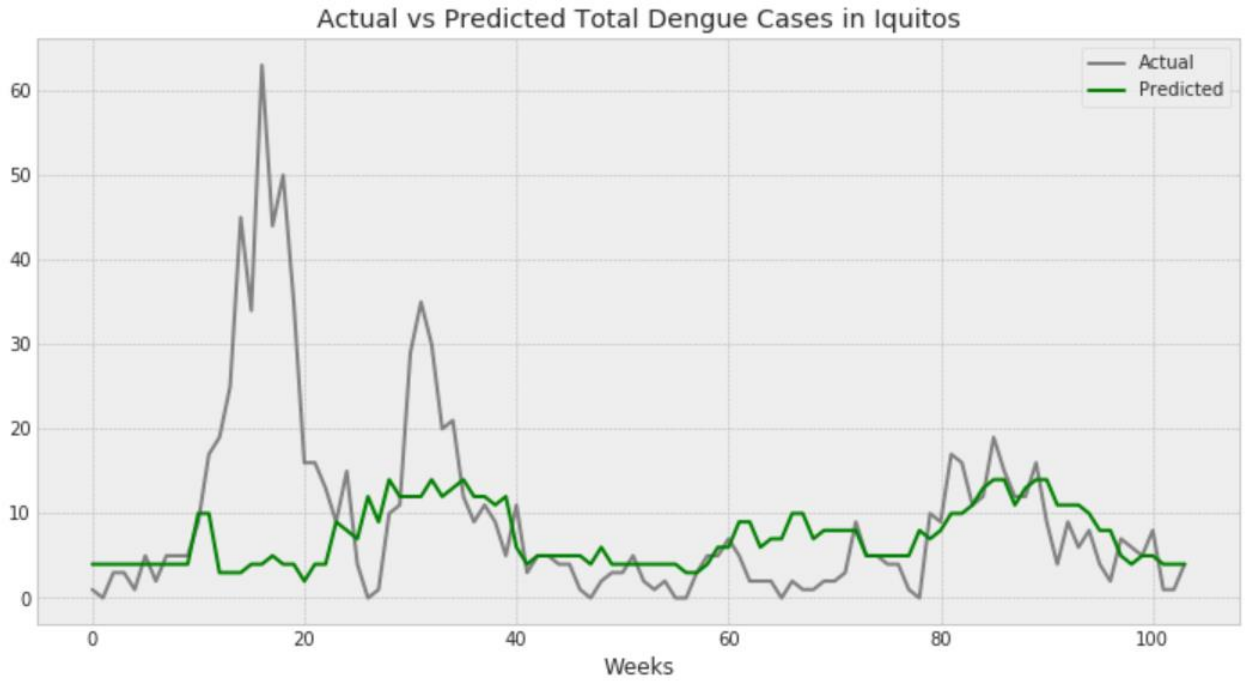


Figure 6 - Actual vs predicted total dengue cases in Iquitos, XGBoost with moving average rolling features added to the model

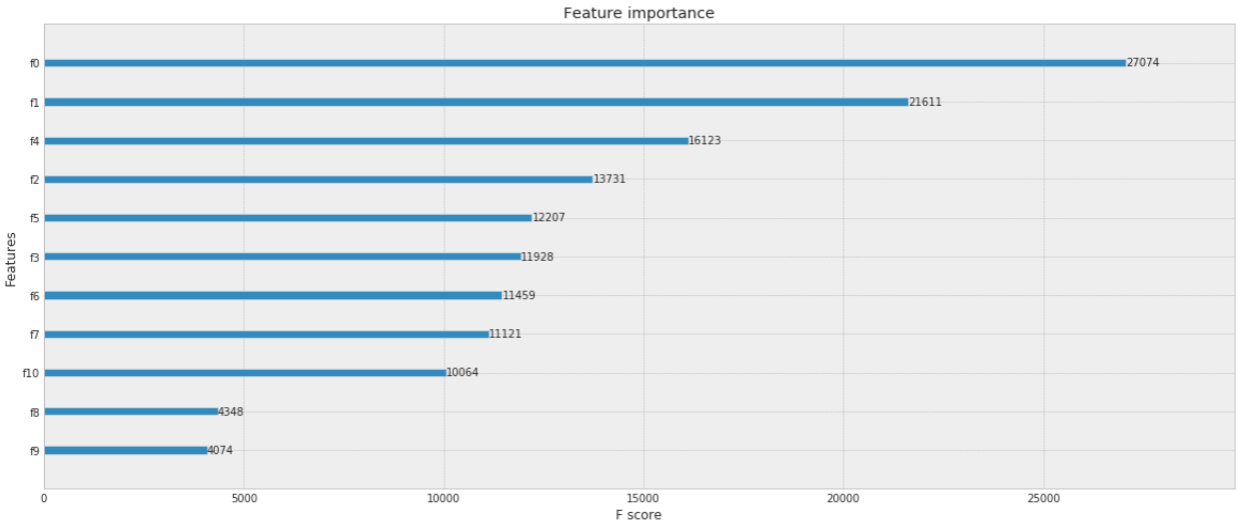


Figure 7 - Feature Importance of XGBoost model for San Juan, most important features are: `ndvi_ne`, `ndvi_nw` *and* `reanalysis_sat_precip_amt_mm`

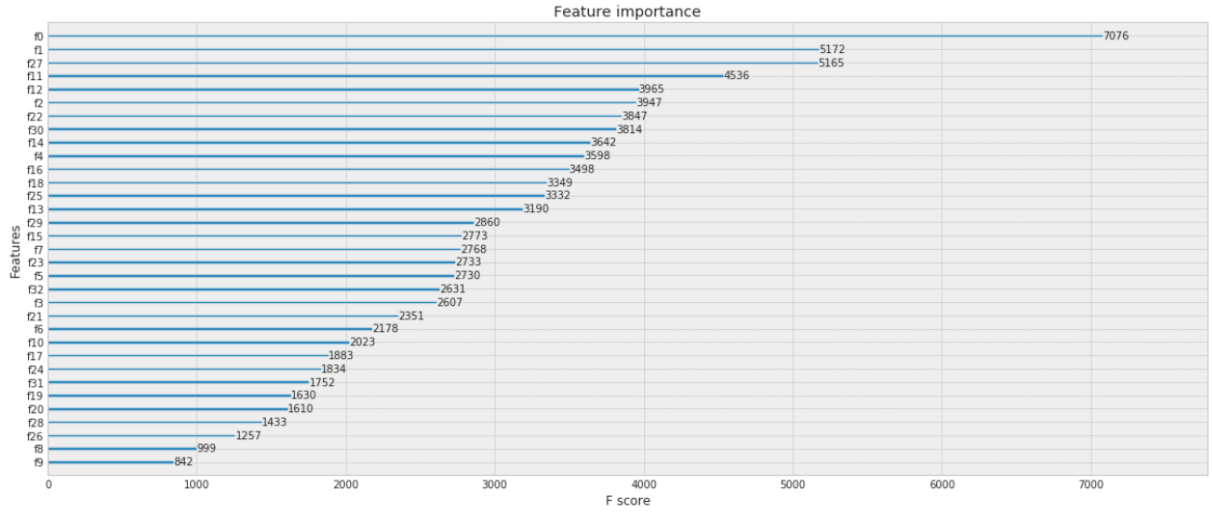


Figure 8 - Feature Importance of XGBoost model for San Juan with moving average rolling features added to the model

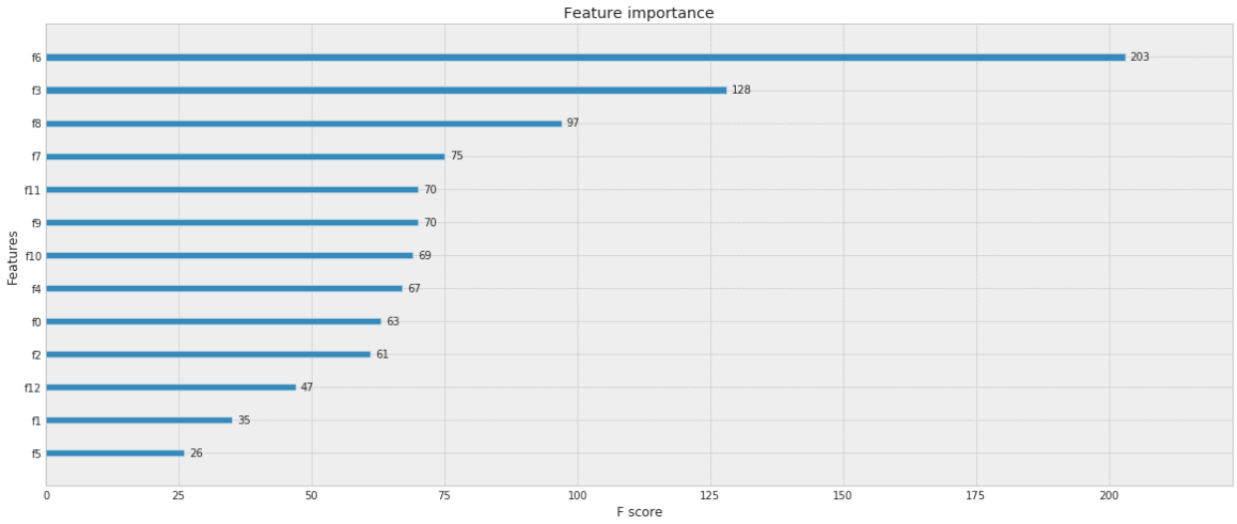


Figure 9 - Feature Importance of XGBoost model for Iquitos, most important features are: `reanalysis_sat_precip_amt_mm`, `ndvi_se` *and* `reanalysis_tdtr_k`

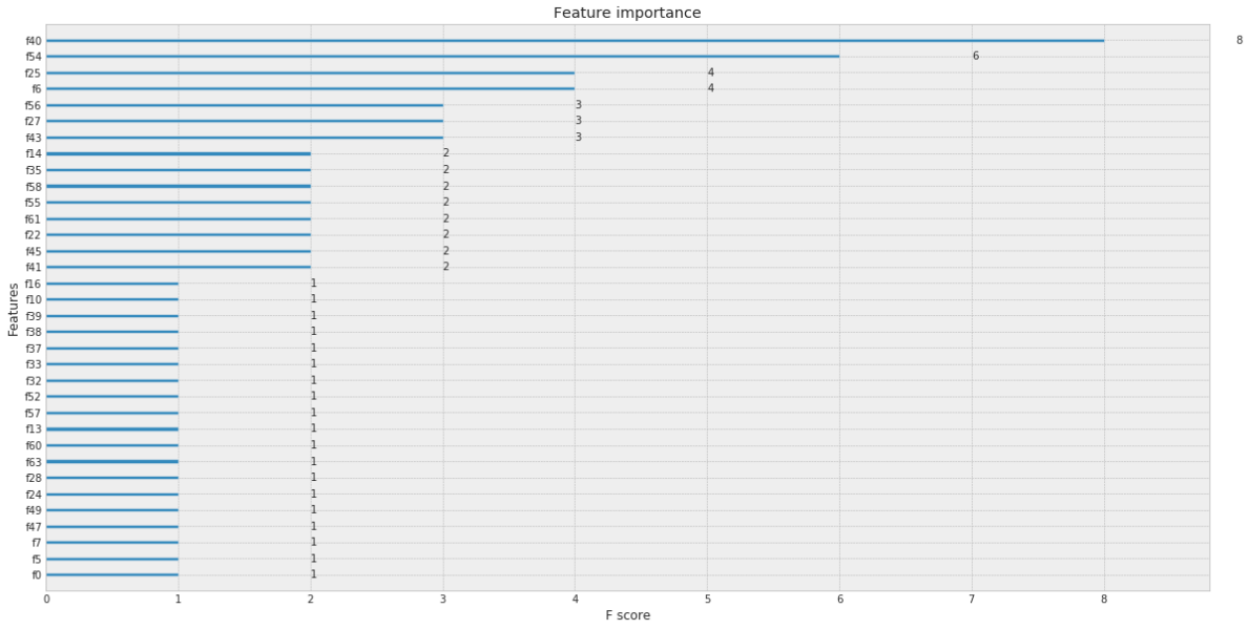


Figure 10 - Feature Importance of XGBoost model for Iquitos with moving average rolling features added to the model

LSTM Implementation Appendix

```
Epoch 250/250
746/746 [==============================] - 2s 3ms/step - loss: 15.3905 - mean_absolute_error: 15.3905 - val_loss: 1
6.5235 - val_mean_absolute_error: 16.5235
```

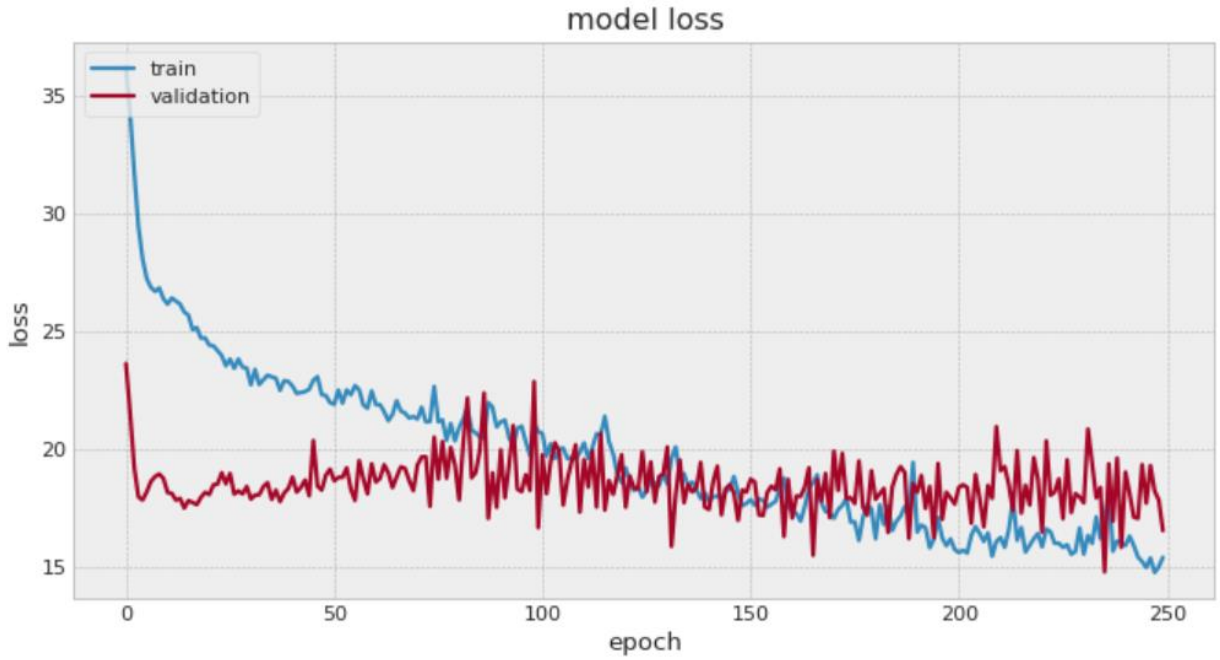


Figure 11 - Model loss plot, training and validation set for San Juan

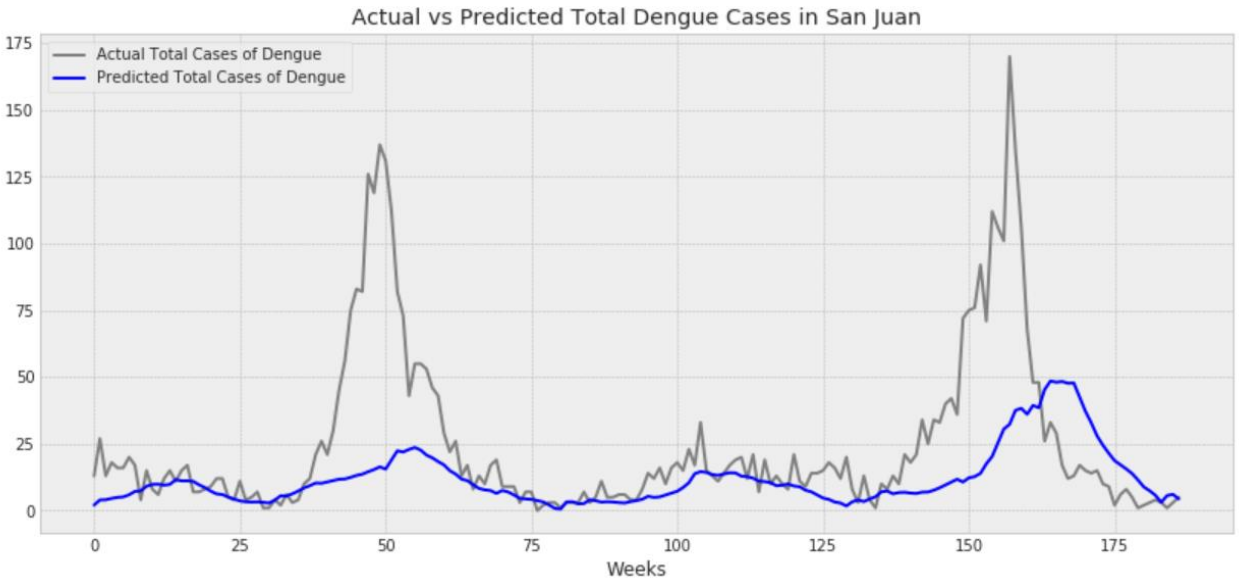


Figure 12 - Actual vs predicted total dengue cases in San Juan, LSTM model

```
Epoch 50/50
310/310 [==============================] - 0s 66us/step - loss: 114.1720 - mean_absolute_error: 6.5458 - val_loss: 1
11.1023 - val_mean_absolute_error: 6.5825
```

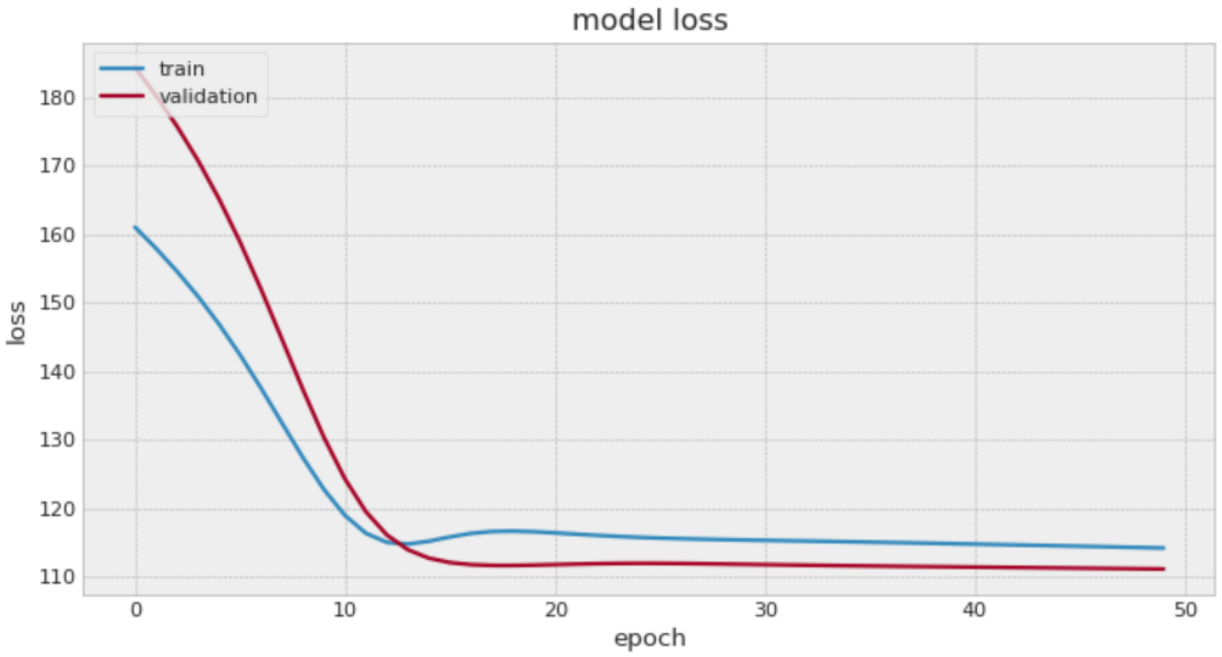


Figure 13 - Model loss plot, training and validation set for Iquitos

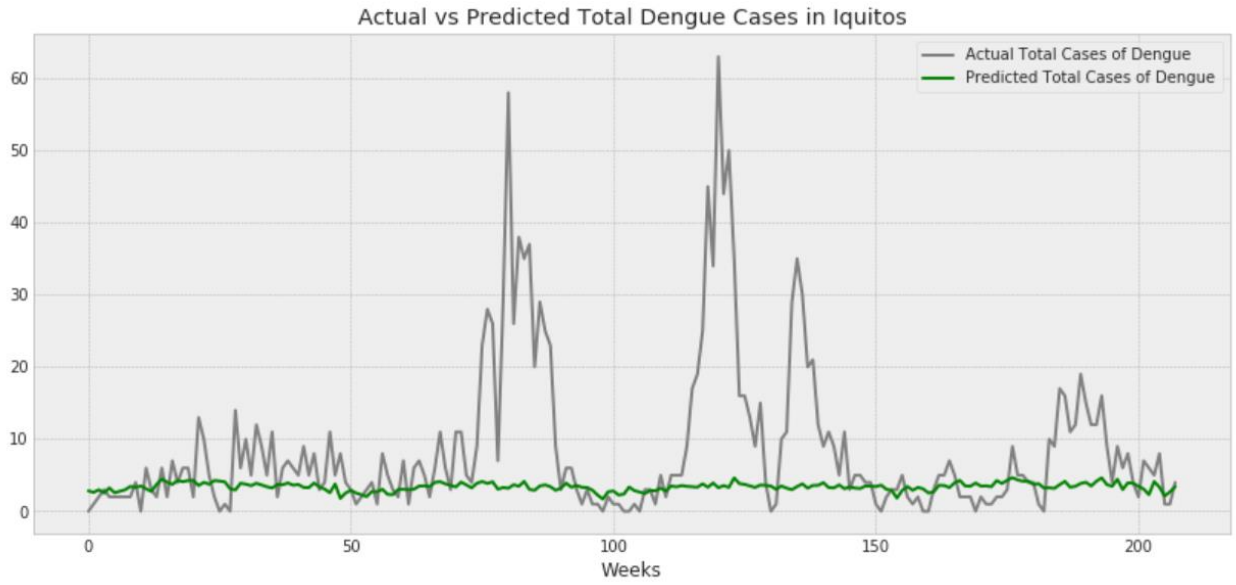


Figure 14 - Figure 12 - Actual vs predicted total dengue cases in Iquitos, LSTM model

Benchmark Model Implementation Appendix

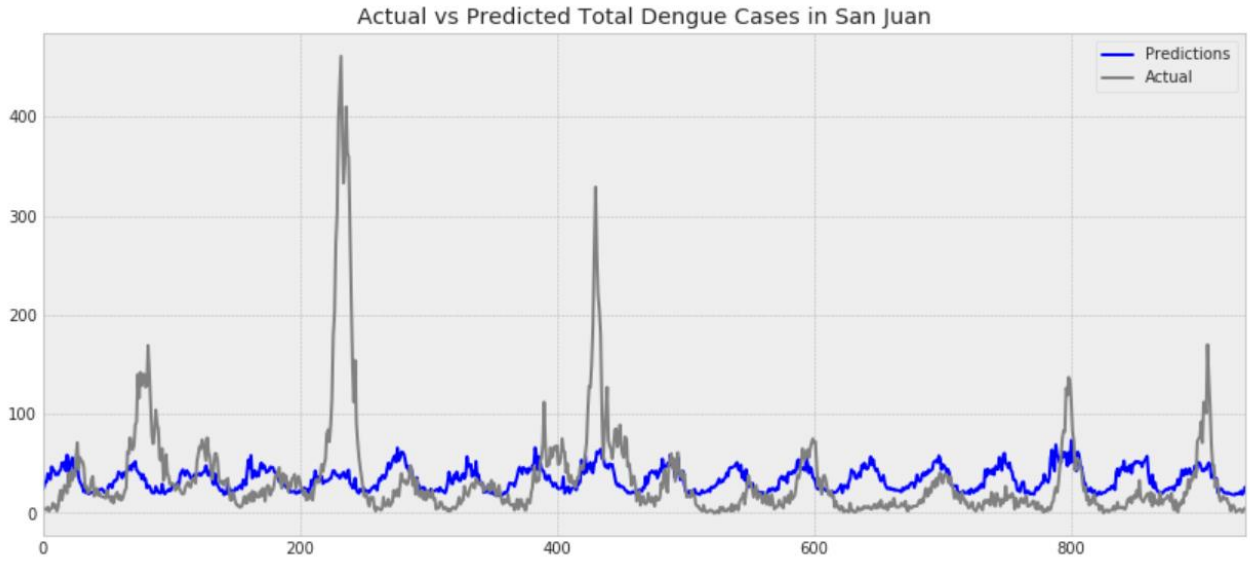


Figure 15 - Actual vs predicted total dengue cases in San Juan, Benchmark Model

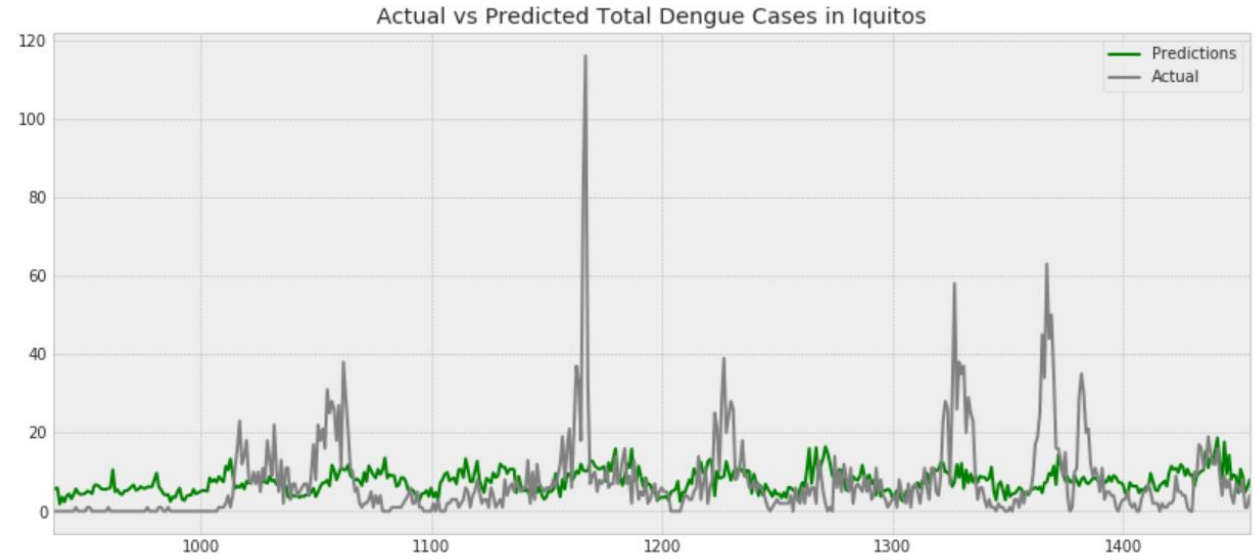


Figure 16 - Actual vs predicted total dengue cases in Iquitos, Benchmark Model

Predictions

San Juan

city	year	weekofyear	XGB	LSTM	NB
sj	2008	18	2.8732831	2.014888	27
sj	2008	19	2.9888713	2.876015	25
sj	2008	20	3.2675917	1.32887	33
sj	2008	21	2.8152888	1.115518	27
sj	2008	22	2.9181366	0.669663	29
sj	2008	23	2.9938614	1.643555	30
sj	2008	24	2.9126165	2.110805	32
sj	2008	25	3.1569054	1.600049	40
sj	2008	26	3.1565812	1.831848	42
sj	2008	27	3.5300968	2.53735	39
sj	2008	28	3.5291271	2.83486	34
sj	2008	29	3.1698608	3.402557	37
sj	2008	30	2.953969	3.600614	44
sj	2008	31	31.25004	4.344535	41
sj	2008	32	62.773354	5.756842	51
sj	2008	33	83.173416	6.649509	52
sj	2008	34	70.34087	8.852572	53
sj	2008	35	74.94514	10.78134	58
sj	2008	36	82.57488	12.87319	71
sj	2008	37	72.87098	16.90606	63

Table 4 - First 20 predictions for San Juan from all the fitted models

Iquitos

city	year	weekofyear	XGB	LSTM	NB
iq	2010	26	0.137978	2.628696	7
iq	2010	27	0.137978	2.519951	5
iq	2010	28	6.496848	2.749619	8
iq	2010	29	0.137978	1.330771	2
iq	2010	30	0.137978	2.201771	2
iq	2010	31	0.137978	2.972654	5
iq	2010	32	0.137978	2.133934	3
iq	2010	33	0.137978	2.496053	5
iq	2010	34	0.137978	2.626015	4
iq	2010	35	0.137978	3.177805	6
iq	2010	36	0.137978	2.334809	6
iq	2010	37	0.137978	2.754367	5
iq	2010	38	0.137978	3.069164	5
iq	2010	39	0.137978	3.07362	5
iq	2010	40	0.499023	3.128445	5
iq	2010	41	0.499023	3.053035	5
iq	2010	42	6.857893	2.786401	6
iq	2010	43	6.857893	3.121947	9
iq	2010	44	6.595395	3.486311	12
iq	2010	45	6.595395	3.493835	7

Table 5 - First 20 predictions for Iquitos from all the fitted models

Python Requirements

```
import pandas as pd
from pandas.api.types import is_numeric_dtype
import numpy as np
from numpy import array
from math import sqrt
from statsmodels.tools import eval_measures
import statsmodels.formula.api as smf
import statsmodels.api as sm
from datetime import *
import seaborn as sns
sns.set_style('darkgrid')
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
%matplotlib inline
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score, KFold
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
from sklearn.preprocessing import MinMaxScaler
# xgboost
!pip install xgboost
import xgboost as xgb
print("xgboost", xgb.__version__)
from xgboost import XGBRegressor
from xgboost import plot_importance
# LSTM
!pip install tensorflow
!pip install keras==2.3.1
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from keras.layers import Bidirectional
%load_ext autoreload
%autoreload 2
```