Fundação Getulio Vargas
Applied Mathematics School

A machine learning approach to Dengue forecasting - Comparing LSTM, Random Forest and Lasso

Elisa Mussumeci

Rio de Janeiro
2018

Elisa Mussumeci

Machine learning aproach for Dengue forecasting - Comparing LSTM, Random Forest and lasso

Dissertação submetida à Escola de Matemática Aplicada como requisito parcial para a obtenção do grau de Mestre em Modelagem Matemática da Informação.

Área de Concentração:

Orientador: Flávio Codeço Coelho

Rio de Janeiro
2018

Ficha catalográfica elaborada pela Biblioteca Mario Henrique Simonsen/FGV

Mussumeci, Elisa
A machine learning approach to dengue forecasting : comparing LSTM,
Random Forest and Lasso / Elisa Mussumeci. - 2018.
58 f.

Dissertação (mestrado) – Fundação Getulio Vargas, Escola de Matemática Aplicada.
Orientador: Flávio Codeço Coelho.
Inclui bibliografia.

1. Análise de séries temporais. 2. Redes neurais (Computação). 3. Modelagem de dados. 4. Análise de regressão. 5. Dengue. I. Coelho, Flávio Codeço, 1969-. II. Fundação Getulio Vargas. Escola de Matemática
Aplicada. III. Título.

CDD – 006.3

ELISA MUSSUMECI BIANOR DOS SANTOS

"MACHINE LEARNING APROACH FOR DENGUE FORECASTING - COMPARING LSTM, RANDOM FOREST AND LASSO METHODS".

Dissertação apresentado(a) ao Curso de Mestrado em Modelagem Aplicada da Informação do(a) Escola de Matemática Aplicada para obtenção do grau de Mestra(a) em Modelagem Matemática da Informação.

Data da defesa: 12/04/2018

ASSINATURA DOS MEMBROS DA BANCA EXAMINADORA

Flávio Codeço Coelho
Orientador(a)

Rodrigo Targino
Membro Interno

Leonardo Soares Bastos
Membro Externo

Acknowledgements

I would like to thank the Applied Mathematics School of Fundação Getulio Vargas for the incredible learning opportunity. Also would like to thank my advisor Flávio Codeço Coelho, and professor Renato Rocha Souza for the attention and patience in those two year of work.

Abstract

We used the Infodengue database of incidence and weather time-series, to train predictive models for the weekly number of cases of dengue in 790 cities of Brazil. To overcome a limitation in the length of time-series available to train the model, we proposed using the time series of epidemiologically similar cities as predictors for the incidence of each city. As Machine Learning-based forecasting models have been used in recent years with reasonable success, in this work we compare three machine learning models: Random Forest, lasso and Long-short term memory neural network in their forecasting performance for all cities monitored by the Infodengue Project.

Contents

1 Introduction **10**

2 Literature Review **13**

2.1 Epidemic forecasting 13

2.1.1 Time Series 13

2.1.2 Forecasting 14

2.2 Hierarchical Clustering 15

2.3 Machine Learning and forecasting 16

2.3.1 Random Forest 19

2.3.2 Lasso 21

2.3.3 Long-short term memory (LSTM) 22

3 Methodology **30**

3.1 Methodology 30

3.1.1 Data sources 30

3.1.2 Data modeling 31

3.1.3 Forecasting 33

3.2 Results 37
3.2.1 Cluster analysis 37
3.2.2 Forecasting 38

4 Conclusions and Final Considerations **54**

References **57**

List of Figures

2.1 Machine Learning algorithms map 18

2.2 Decision Tree example 19

2.3 Example of a artificial neural network structure. 23

2.4 Example of a recurrent neural network. 24

2.5 From RNN to LSTM. 25

2.6 Cell state information and gate example. 25

2.7 LSTM first gate. 26

2.8 LSTM second gate. 27

2.9 Update C_t cell state. 28

2.10 LSTM third gate. 28

3.1 Time series from Rio de Janeiro features 31

3.2 Correlation distance matrix of cities for the state of Rio de Janeiro 32

3.3 Clustered features data. 34

3.4 Feature matrix transformation for Random Forest input. . . . 36

3.5 Creation of features matrix (tensors) used to train the LSTM model. 41
3.6 Matrix colormap of dengue incidence for some clusters from Rio de Janeiro state. 42
3.7 Hierarchical cluster of Rio de Janeiro. 43
3.8 Map of Hierarchical clusters for each state. 44
3.9 Loss comparison between models trained with and without cluster features for Rio de Janeiro state. 45
3.10 Loss comparison between models trained with and without cluster features for Ceará state. 46
3.11 Loss comparison between models trained with and without cluster features for Paraná state. 47
3.12 Count of errors for each model. 48
3.13 QQplot of errors for each state. 49
3.14 MASE loss function analysis. 50
3.15 Forecasting for cities of Ceará. 51
3.16 Forecasting for cities of Rio de Janeiro. 52
3.17 Forecasting for cities of Paraná. 53

Chapter 1

Introduction

Dengue fever is a mosquito-borne viral disease that has a quick spread and affects many countries, especially Asian and Latin American countries. It has become a leading cause of hospitalization and death among children and adults in these regions.

The incidence of dengue has grown dramatically around the world in recent decades, impacting the health, social, and economies of many countries. That makes predictive models for epidemiological time series very relevant, since it can help health authorities and population to prepare for a potential epidemic (Luz et al., 2008).

Understanding and therefore being able to predict the incidence of vector-borne diseases is a big challenge due in part to the complex interplay between epidemiological and environmental determinants but also due to the frequent lack of long-term records of historical disease incidence and other cofactors

affecting risk. This complex dynamics manifests itself in the variability of the incidence patterns in different geographical areas.

In the case of dengue, the effects of weather of the vector's population dynamics impose a marked seasonality which is then modulated by variations in the immunological structure of the population as well as by human demography (Stoddard et al., 2013).

Modeling dengue incidence patterns in Brazil presents an additional challenge characterized by the fact that the disease was reintroduced in the country in the 1980s having been absent for many decades. Therefore, besides the problem of short time series we are dealing with a disease which has yet to reach its endemic equilibrium. Any forecasting model hoping to grasp the complex temporal dynamics of dengue will require a large amount of data.

In this paper, we use the Infodengue (Codeco et al., 2016) database of more than 700 cities in Brasil with data starting from 2010. To make up for the relatively short time series, we use use incidence series from epidemiologically similar cities as predictors, together with relevant environmental time series for each city to fit predictive models capable of forecasting the weekly incidence of Dengue for any city of Brazil across a wide range of latitudes and climate characteristics.

Instead of proposing parsimonious models built from previous knowledge of the determinants of dengue transmission, we fit machine learning models known for their ability to navigate their way into data-intensive high dimensional problems, by integrating variable selection into their fitting routine.

Namely, we will compare the following models: LSTM, a recurrent deep neural network model, Random Forest regression and lasso regression.

Chapter 2

Literature Review

2.1 Epidemic forecasting

The spread of a infectious disease to a large number of people within a short period of time is denominated an *epidemic*. To predict if a epidemic is going to occur, we observe the incidence of this disease through the past, and also the features that may have a correlation that can help to explain this epidemic.

When looking to an incidence, or any other type of value, through time, we are looking at time series.

2.1.1 Time Series

A time series is a series of data points observed through time. It is an extensively studied theme (Box et al., 2015; Tong, 1990; Scharf and Demeure,

1991), that can lead to important results, especially when dealing with predictions.

Time series analysis comprises methods for analyzing the observed data in order to extract meaningful statistics and new information. Observing some characteristics from time series, such as the seasonality, trends and cycles, helps to understand the data, and therefore to choose the best technique to apply in order to accurate predictions.

2.1.2 Forecasting

Time series forecasting consists in the study of complex models to predict future values based on previously observed values.

To study the performance of a model, the forecast error (residual) is calculated. It consists in the difference between the predicted and empirical value as show in equation (2.1)

$$E_t = Y_t - \hat{Y}_t \tag{2.1}$$

where E_t is the forecast error at time t, Y_t is the real value of the series at time t and $\hat{Y}_t$ is the predicted value at time t. In table 2.1 there are some measures of performance calculated from E_t.

Name	Equation
MAE	$\frac{\sum_{t=1}^{N} \lvert E_t \rvert}{N}$
MSE	$\frac{\sum_{t=1}^{N} E_t^2}{N}$
MAPE	$\frac{100}{N} \sum_{t=1}^{N} \frac{\lvert E_t \rvert}{\lvert Y_t \rvert}$
MASE*	$\frac{1}{N} \sum_{t=1}^{N} \frac{E_t}{\frac{1}{N-m} \sum_{t=m+1}^{N} \lvert Y_t - Y_{t-m} \rvert}$

Table 2.1: Measures to calculate loss. *m is the seasonal period; $m = 1$ if non-seasonal.

2.2 Hierarchical Clustering

Clustering is a statistical technique for data analysis, extremely useful in exploratory data analysis (Anderberg, 1973). The main goal of such technique is to group a set of elements into subsets of similar elements. Each observation must be more similar to those of the same subset than to others in different subsets.

Cluster analysis itself is not one specific algorithm, but the general task to be solved. For each specific data set or expected output there'll be a more efficient clustering algorithm to be applied, therefore, clustering can be formulated as a multi-objective optimization problem.

There are several cluster models, such as connectivity models, centroid models, distribution models, etc (Estivill-Castro, 2002). In this work we'll focus on the hierarchical clustering, a connectivity model.

Hierarchical clustering is a method of cluster analysis which builds a

hierarchy between its clusters. The two mostly applied algorithms to build hierarchical clusters are the agglomerative and the divisive (Rokach and Maimon, 2005). The main difference between them is that the the agglomerative starts with each elements being it's own cluster and then pairs with another's until all of them are connected (bottom up approach); on the divisive, all elements starts as a single cluster which is recursively split down into the hierarchy (top down approach).

To define if a set of elements must be clustered, the algorithm base it's decision into an appropriate *metric*. This metric calculates the distance between any two elements, and will influence the shape of the clusters, as some elements may be close to one another according to one distance metric and farther away according to another. Table 2.2 shows some distance metrics.

The *linkage* criteria (table 2.3) determines the distance between clustered sets as a function of the pairwise distances between the elements of each set. If this distance between two clusters is less than some predefined threshold, than those clusters are considered brothers in the hierarchy.

2.3 Machine Learning and forecasting

Machine learning is a field of computer science whose goal is to devise complex models and algorithms that can learn from data and lend themselves to prediction. There are several techniques and algorithms inside this field that deals with traditional problems such as classification, regression and

Distance	**Equation**
Euclidean	$\|u - v\|_2$
Correlation	$1 - \frac{(u - \bar{u}) \cdot (v - \bar{v})}{\|(u - \bar{u})\|_2 \|v - \bar{v}\|_2}$
Chebyshev	$\max \|u_i - v_i\|$
Mahalanobis*	$\sqrt{(u - v)V^{-1}(u - v)^T}$

Table 2.2: Some metrics used for calculate distance between the vector u and v. *V is the covariance matrix.

Method	**Equation**
Single	$min(dist(u[i], v[j]))$
Complete	$max(dist(u[i], v[j]))$
Centroid	$\|c_s - c_t\|_2$
Average	$\sum_{ij} \frac{d(u[i], v[j])}{(\|u\| * \|v\|)}$

Table 2.3: Linkage methods used for calculating distances between the clusters u and v. $d(u[i], v[j])$ is the distance between all points i in cluster u and j in cluster v. c_s and c_t are the centroids of clusters s and t.

clustering. In figure 2.1 we have a *machine learning algorithm map*[1] with some of the most famous algorithms.

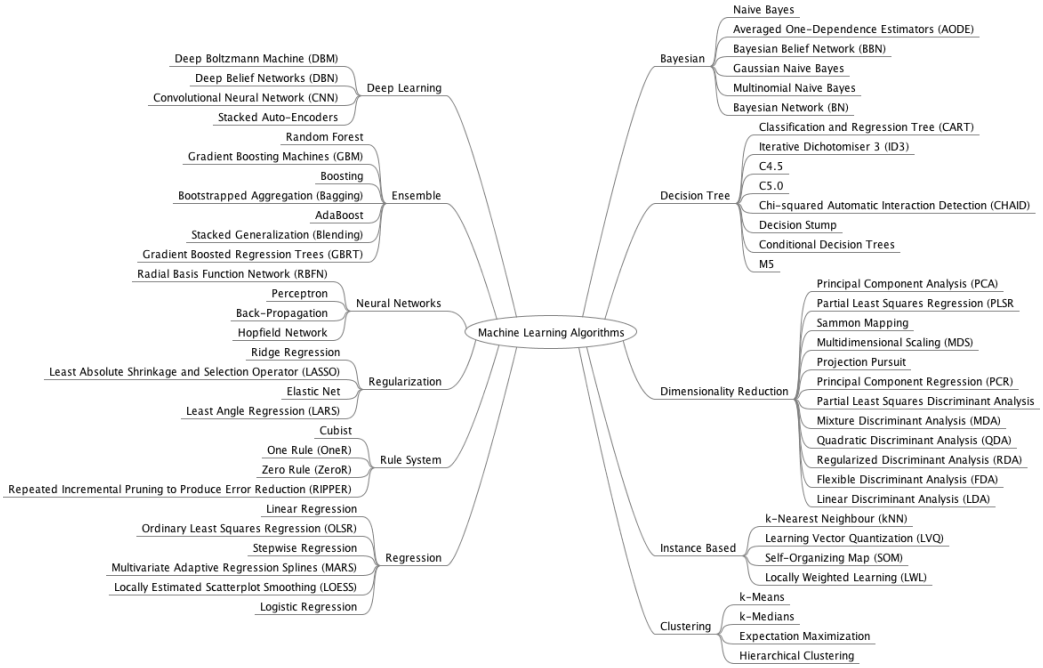

Figure 2.1: Machine Learning algorithms map

Machine learning techniques can also be useful to forecasting problems, as shown in Shen et al. (2012), Santillana et al. (2015) and Bai and Jin (2005).

In this article we'll approach three machine learning methods: Random Forest, lasso and LSTM.

[1]source https://machinelearningmastery.com/

2.3.1 Random Forest

Decision Trees

Decision Trees are a type of supervised learning method that builds classification and regression models applying a tree structure. The model learns decision rules that are used to predict the values of the target variable.

The decision tree is created by splitting the dataset into smaller homogeneous subsets and associating them based on some inferred rule. In figure 2.3.1 we have an example of this construction.

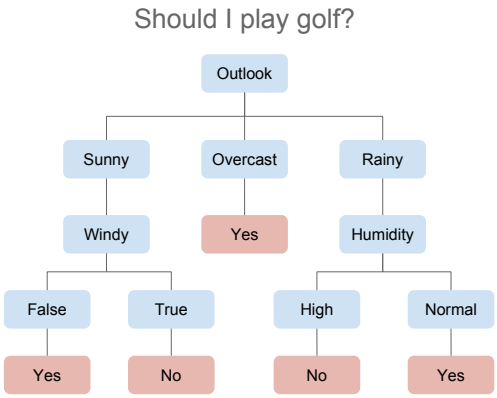

Figure 2.2: Decision Tree example

Decision Trees are extremely fast to fit and easy to interpret; they perform well on large datasets and can handle numerical and categorical data. The deeper the tree, the more complex the decision rules are and more accurate is the model. The one big disadvantage of decision trees is that they are prone to overfitting.

The deeper the tree, the smaller the subsets, which will result in a very

accurate model only for the specific training set, that results in an overfitted model. One way to combat this issue is by setting a maximum depth, which would limit our risk of overfitting; but, at the expense of error due to bias and reduced predictive power.

The answer to minimize both error due to bias and error due to variance is to upgrade our Decision Tree to a Random Forest model.

Random Forest

Random Forests (Breiman, 2001) are an learning method for classification and regression build from the ensemble of multiple decision trees. The ability to limit overfitting without substantially increasing error due to bias is what makes Random Forest such a powerful method.

The Random Forest algorithm applies a bagging method to the decision trees, which consist of selecting N random samples with replacement from the training set and fitting a modified tree learning algorithm to each one of them.

This modified tree learning algorithm instead of looking through all features in order to select the most optimal split-point, is limited to a random subset of features to choose. This process, usually called *feature bagging*, allows the sub-trees to have minor correlation between them, which increases the performance of the bagging method since combining predictions from multiple models in ensembles works better if the predictions from the sub-models are weakly correlated.

The number of features p that can be searched at each split point is specified as a parameter to the algorithm. This value can be tuned it using cross validation and the default is usually set to $p/3$ for regression models and $\sqrt{p}$ for classification models.

Given the data $D = \{(X_1, y_1), (X_2, y_2), ..., (X_n, y_n)\}$, where X_n is the feature vector and y_n is the target value, we define $h = \{h(X|\theta_1), h(X|\theta_2), ..., h(X|\theta_k)\}$ as the set of decision trees to ensemble, where $X = X_1, X_2, ..., X_n$ and θ_k is the vector of p features randomly chosen.

Our random forest predictor $f(X)$ is defined by:

$$f(X) = \frac{1}{k} \sum_{i=1}^{k} h(X|\theta_i)$$

The uncertainty of the prediction is given by the standard deviation:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{k} (h(X|\theta_i) - f(X))^2}{k-1}}$$

2.3.2 Lasso

Lasso regression analysis (Least Absolute Shrinkage and Selection Operator) is a shrinkage and variable selection method for linear regression models. By imposing a constraint on the model parameters it causes regression coefficients for some variables to shrink toward zero, with the goal to find a subset of predictors that minimizes prediction error for a quantitative response variable.

Lasso regression performs ℓ_1 regularization, which adds a penalty equal to the absolute value of the magnitude of coefficients. The objective function to minimize is given by equation 2.2.

$$\underset{\beta \in R^p}{min} \frac{1}{N} ||y - X\beta||_2^2 + \lambda ||\beta||_1 \tag{2.2}$$

,

The lasso estimate solves the minimization of the least-squares penalty with $\lambda||\beta||_1$ added, where $||\beta||_1$ is the ℓ_1-norm of the parameter vector and λ is a constant that controls the degree of sparsity of the coefficients estimated, that means, controls the strength of the ℓ_1 penalty.

When $\lambda = 0$, no parameters are eliminated. The estimate is equivalent to an ordinary least square, equal to the one found with linear regression. As λ increases, more and more coefficients are set to zero and eliminated and the bias increases. When λ decreases, variance increases.

2.3.3 Long-short term memory (LSTM)

Artificial Neural Networks

Artificial Neural Networks (ANN) are collections of connected units called artificial neurons. Each connection between artificial neurons can transmit a signal from one to another, usually this signal is a real number, and the output of each neuron is calculated by a non-linear function of the sum of its inputs. Those connections have weights that increase or decrease the

strength of the signal and are adjusted at the learning process.

Neurons are organized in layers, where each layer may apply a different kind of transformation (activation function) to its inputs. Figure 2.3 show an example of the structure of a neural network.

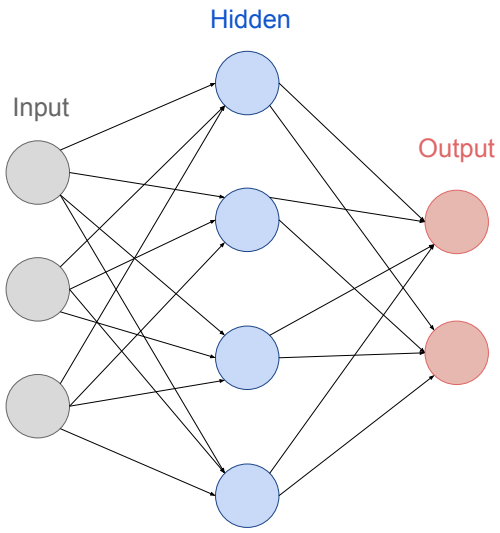

Figure 2.3: Example of a artificial neural network structure. Each circle is a neuron, they are organized as layers. In this example we have a input layer with three neurons, one hidden layer with four neurons and the output layer with two neurons.

Recurrent Neural Networks

Recurrent neural networks (RNN) are neural networks with loops in them, allowing information to persist. They can be thought of as multiple copies of the same network, each passing a message to a successor. In the figure 2.4 we have the structure of a RNN, where each cell A is the chunk of a neural network, x_t is the cell input and h_t is the cell output.

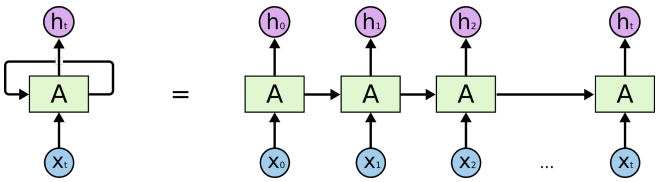

Figure 2.4: Example of a unrolled recurrent neural network (Olah, 2015).

RNN's learns to use the past information, that means, they remember the information given at time t_0 when predicting time t_τ. They usually have great performance when the gap between $t_\tau - t_0$ is small; but they can't handle long-term dependencies. So as $t_\tau - t_0$ increases (the gab between the useful past information and the time we want to predict increase), the network becomes unable to learn how to connect the information, which in theory should be able. This problem was extensively explored by Bengio et al. (1994).

Long-short term memory

Long-short Term Memory Networks (Hochreiter and Schmidhuber, 1997) are a special kind of RNN, capable of learning long-term dependencies. Figure 2.5 shows the difference between those two networks.

In each cell of the chain the network decides which informations to keep and which to forget.

Two important elements from the LSTM cell are the cell state and the gates. The cell state C_t, makes it possible for information to flow through the cell unchanged or with some minor interactions.The gates are responsible

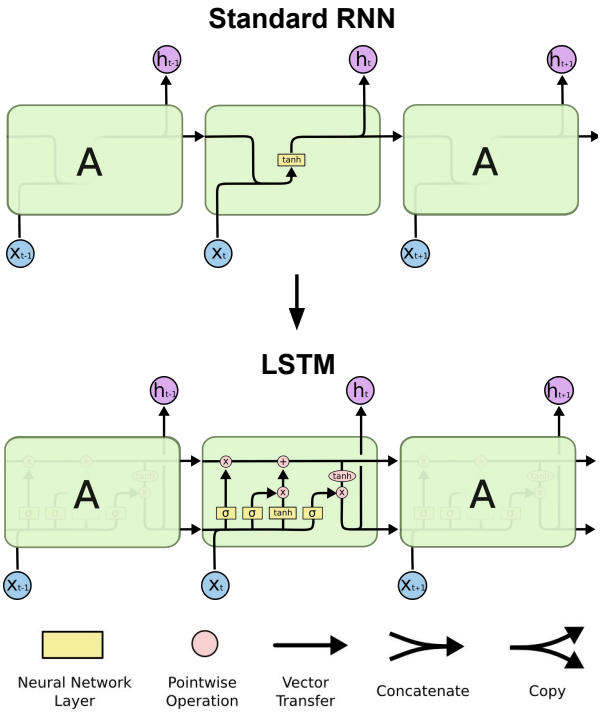

Figure 2.5: From RNN to LSTM (Olah, 2015). The standard RNN network has in each cell only one gate with its activation function. The LSTM has a more complex structure with four gates that allows information to persist.

for those interactions by protecting and controlling the cell state, as shown in figure 2.6.

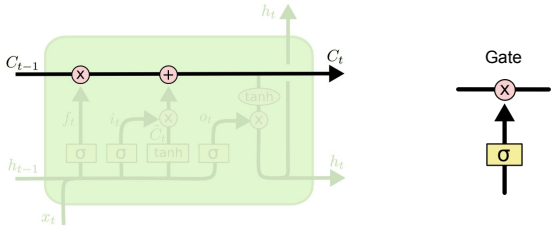

Figure 2.6: Cell state information and gate example (Olah, 2015).

Each LSTM cell has 3 gates, composed by a sigmoid neural net layer and a

pointwise multiplication operation. The sigmoid layer tell the cell how much information must be kept and how much must go; it takes values between 0 and 1 and it is described by the sigmoid function in (2.3).

$$S(Z) = \frac{1}{1 + e^{-z}} \tag{2.3}$$

The first gate to pass information to the cell state is the *forget gate layer* which is composed by a sigmoid layer (σ) and decides which information to forget from the previous cell state C_{t-1}. It has as input the previous output value h_{t-1} and a new information x_t and output a value f_t between 0 and 1. Equation 2.4 shows this procedure, where W_f is the weights vector of this gate.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.4}$$

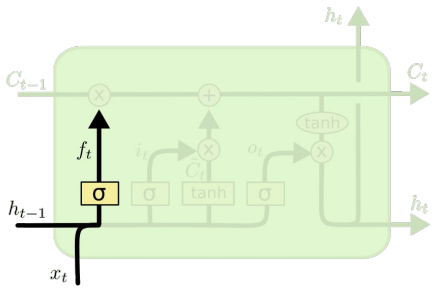

Figure 2.7: LSTM first gate (Olah, 2015).

The second gate, is composed by the combination of a sigmoid layer called *input gate layer* and a tanh layer, as shown in figure 2.8. This gate decides

which new information we'll update to the cell state; the sigmoid layer (σ) chooses which previous values will be updated, and the tanh layer creates the vector $\tilde{C}_t$, which contains the candidate values to be added to the state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.5}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.6}$$

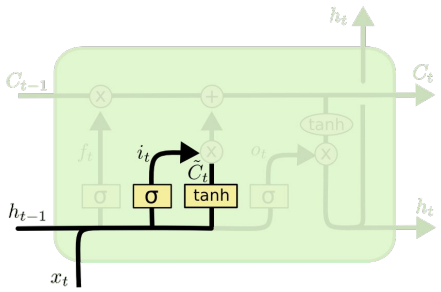

Figure 2.8: LSTM second gate (Olah, 2015).

With equations (2.4), (2.5) and (2.6) defined, we can update the cell state C_{t-1} to C_t. The first thing to do is to forget the chosen information f_t by multiplying it to C_{t-1}. Than we add the combination $i_t * \tilde{C}_t$ which is the new candidate values ($\tilde{C}_t$), scaled by how much we decided to update each state value (i_t):

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.7}$$

The third gate defines the output h_t of the cell. We apply a tanh function

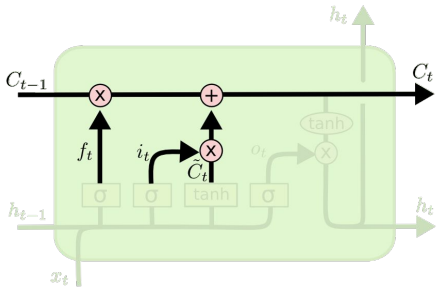

Figure 2.9: Update C_t cell state (Olah, 2015).

to the cell state to push the values between -1 and 1, and multiply by the output of a sigmoid layer, which filters the information from the cell state that is interesting for this h_t.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.8}$$

$$h_t = o_t * tanh(C_t) \tag{2.9}$$

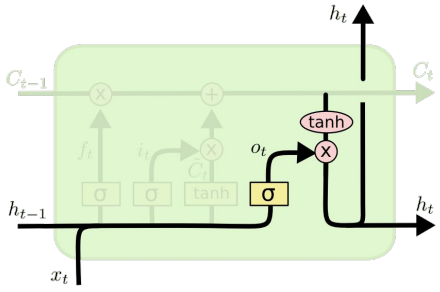

Figure 2.10: LSTM third gate (Olah, 2015).

The cells from the LSTM basically receives a previous state and a new input, choose which information to storage for futures outputs, and which

information to output in this state. This makes possible for the LSTM to learn which information to storage.

Chapter 3

Methodology

3.1 Methodology

3.1.1 Data sources

The dataset used in the article was provided by the InfoDengue project.

InfoDengue Codeco et al. (2016) is an integrated dengue alert system, developed as a partnership between Oswaldo Cruz Foundation and Getulio Vargas Foundation. As a unique source of carefully curated data for epidemiological studies, InfoDengue monitors, until 2018, 790 cities in Brazil, predicting the weekly number of new cases of dengue in each one of them.

The data consist of the weekly series of dengue incidence, temperature, humidity, pressure, precipitation intensity, about dengue in each observed city. It ranges from 2010 until the end of 2017. Figure 3.1 shows an example of this data for the city of Rio de Janeiro, RJ.

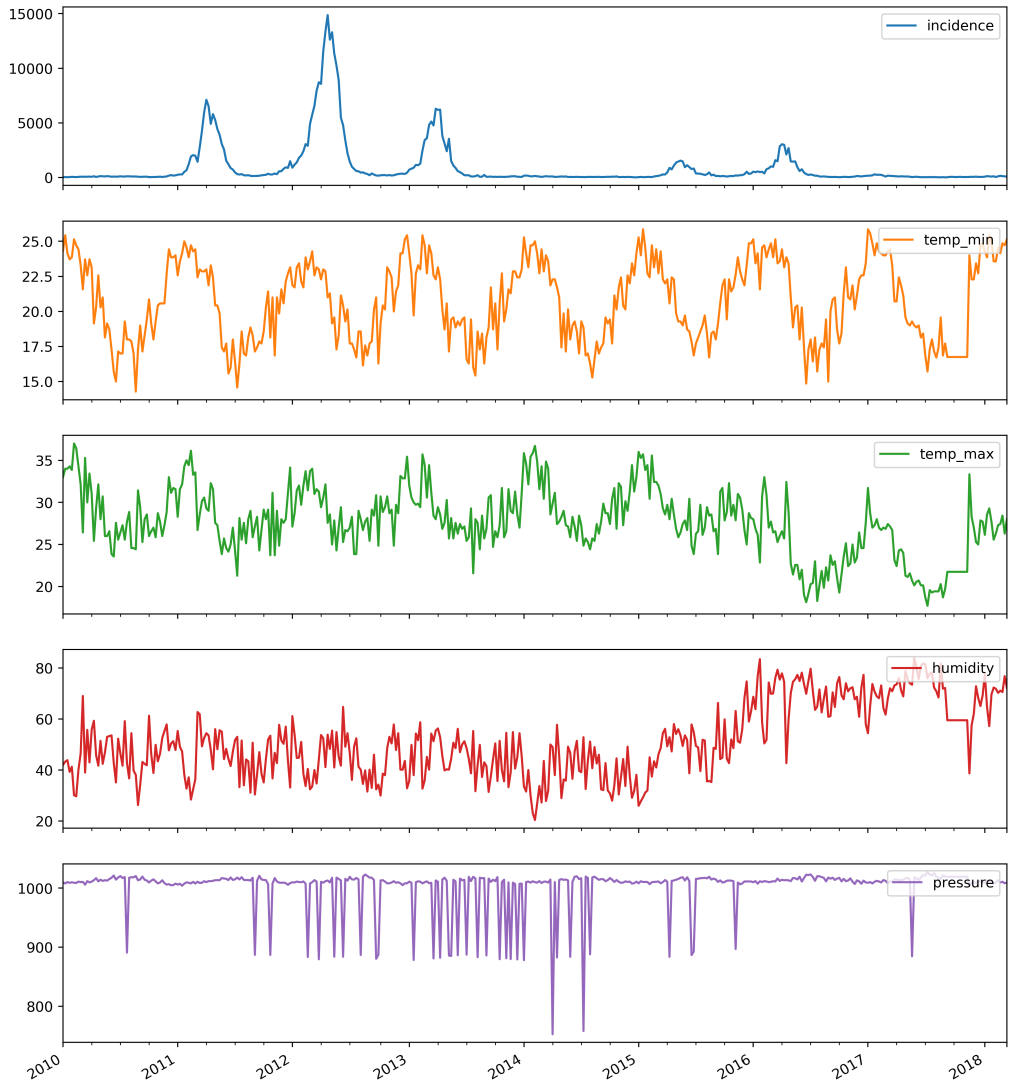

Figure 3.1: Time series from Rio de Janeiro features

3.1.2 Data modeling

It is very difficult to accurately forecast the weekly incidence in a city based only on its historical data. This happens mainly because of the spatial com-

ponent of disease transmission. Diseases perpetuate themselves by moving from population to population and the flow of individuals between cities is also an important factor.

Dengue is a disease with a clear spatial dependency (Stoddard et al., 2013; Eisen and Lozano-Fuentes, 2009). Therefore it makes sense to use series from neighboring cities nearby in order to reduce the uncertainty in its forecasts. Additionally, other cities not necessarily in the vicinity but which display similar historical series of incidence, can also be included as predictors.

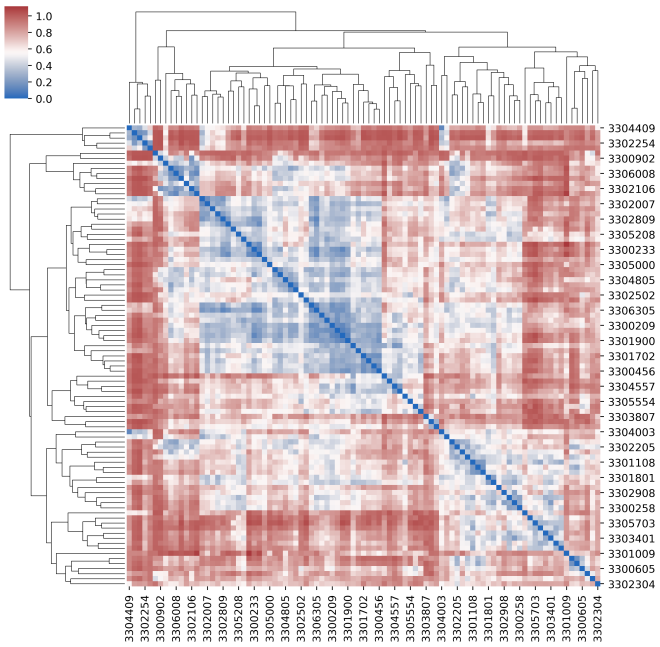

Figure 3.2: Correlation distance matrix of cities for the state of Rio de Janeiro. The cities are represented by its geocodes.

In order to define the set of cities with relevant predictive information for each city, we clustered all cities within a state based on the correlation

distances between incidence time series for each pair of cities. The clusters were calculated applying a hierarchical agglomerative algorithm, where the distance d of each pair of clusters (u, v) was given by the Farthest Point Algorithm:

$$d(u, v) = max(dist(u[i], v[j])),$$

for all points i in cluster u and j in cluster v. The threshold that defines how big this distance can be in a same hierarchical cluster is set to $0.6 * max(z)$, where z is the vector of the pairwise correlation distances of the cities. We can see in Figure 3.2 the correlation matrix for the cities from the state of Rio de Janeiro, the threshold defines the distance between the clusters in Figure 3.7.

For each city, a feature matrix was assembled from the set of time series of each other cities from its cluster (Figure 3.3)

A single city model was fitted for a few cities to serve as a baseline against which to compare the effectiveness of the using a cluster of cities as predictors.

3.1.3 Forecasting

We fit three different classes of models to forecast the incidence time series. For all models we adopted a forecast window of 4 weeks, meaning that from any moment in time the model will generate forecasts for the next 4 weeks based only on historical data up to that point.

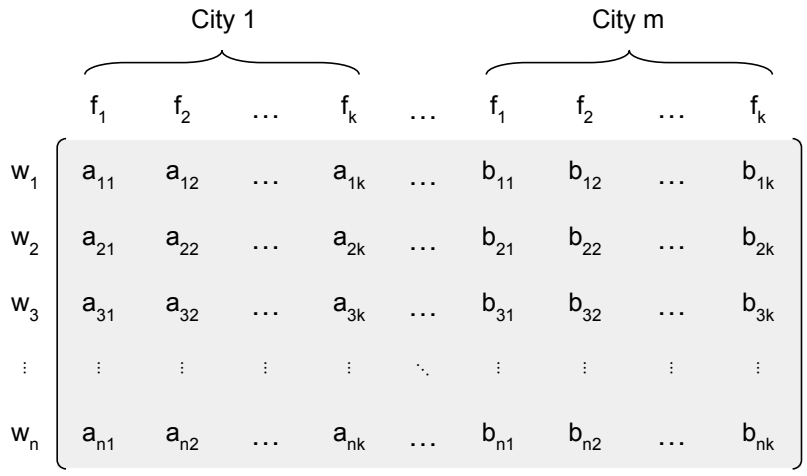

Figure 3.3: Clustered features data with w_n as the total number of week observation, m the total number of cities and f_k the total number of features for each city.

Random Forest:

We used a Random Forest regression model to predict a single point in the future based on historical data. In order to turn a time series prediction problem into a regression model, we transformed the series regressors (matrix in Figure 3.3) into a vector containing not just the last observation of each series but the D most recent.

$$\hat{y}_{t+\tau} = \beta_t T_t \tag{3.1}$$

where t=D,...,n. T_t is defined as $T_t = [X_t, X_{t-1}, ..., X_{t-D}]$, where X_{t-d} is the vector with the values of all m predictor series at time $t - d$, where $d = 0, ..., D$. β_t is a vector $1 \times m$ which contains the weights for

Layer	Output	Activation	Dropout
LSTM	(1,4,4)	tanh	0.2
LSTM	(1,4,4)	tanh	0.2
LSTM	(1,4)	tanh	0.2
Dense	(1,4)	relu	-

Table 3.1: Parameters used to build LSTM model

each value of T_t. The model predicts the incidence at a particular week in the future $y_{t+\tau}$, thus since we wanted to predict 4 weeks into the future, 4 separate models were fitted to data for each τ varying from 1 to 4.

Figure 3.4 exemplify the data transformation for the input of the model described above:

Long short term memory (LSTM)

A LSTM model is a recurrent neural network model developed to handle predictions of time series. We used a LSTM model with topology given in table 3.1. The model was trained for 100 epochs with a look back of 4 weeks and a forecasting window of also 4 weeks. Each layer has 10 hidden layers. The loss function used was the mean squared log error (MSLE) defined in equation (3.2), where $y, \hat{y}$ are the true and predicted value, respectively:

$$MSLE = \frac{1}{n}\sum_{i=1}^{i=n}(log(y_i + 1) - log(\hat{y}_i + 1))^2 \tag{3.2}$$

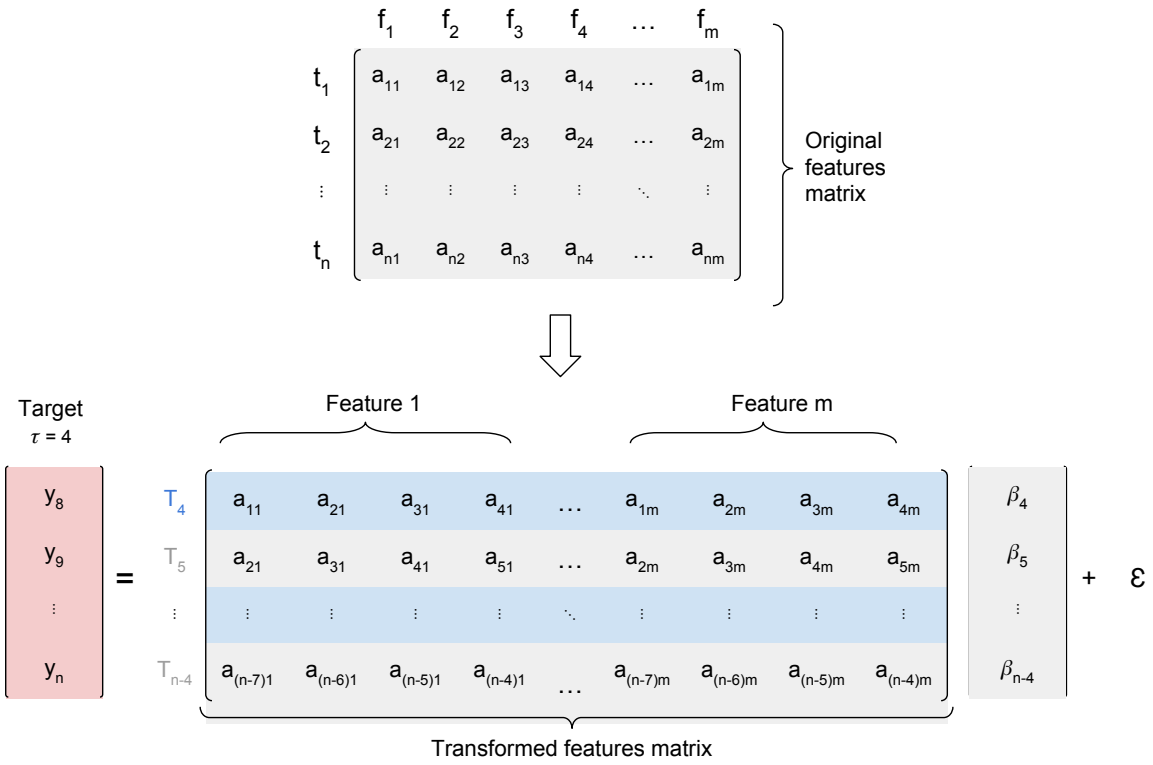

Figure 3.4: Feature matrix transformation for Random Forest input. The first matrix is the original data, where a_{ij} is the feature i at time j, $j = [1, .., n]$, and y_i is the incidence of dengue cases at time i, $i = [1, ..., n]$. In this example, $D = 4$ and $\tau = 4$, so our first vector is T_4, which contains all m predictors series at times (t_1, t_2, t_3, t_4), and our target is $y_{t+\tau} = y_{D+\tau} = y_8$. The last input must contain the last target, therefore $y_{t+\tau} = y_n$ and $T_t = T_{n-4}$.

The feature matrix used to train the LSTM needs to be a 3 dimensional matrix, where each 'slice' is a temporal window from our features regressors (3.3). The first dimension of the matrix is the number of features, therefore has size equals to k; the second dimension is the

number of samples that the model will use to train (total number observations - (look back + forecasting window)); the third dimension is the temporal windows cut observed (look back + forecasting window).

The figure 3.5 exemplifies the construction of this matrix when forecasting window equal to 4 and look back equal to 4.

lasso regression

A lasso model is an estimator for regression models which also performs variable selection as it can estimate sparse coefficient matrices.

We fitted a cross-validation lasso model using the Least Angle Regression (LARS) defined by Efron et al. (2004); additionally, we applied an evolutionary tree-based optimization procedure proposed by Olson et al. (2016) to define the final model.

The feature matrix used was the same used by the Random Forest model (figure 3.4).

3.2 Results

3.2.1 Cluster analysis

We performed the clustering of cities as described in section 3.1.2 for every state in the Infodengue dataset. Figure 3.6 shows the incidence series for a few clusters for Rio de Janeiro state of the entire historical period available.

	Number of cities	Number of Clusters	Cluster mean size
Rio de Janeiro	92	14	6
Paraná	399	54	6
Ceará	184	49	3

Table 3.2: Clusters results comparison between states.

We can see that cities clustered together display similar incidence patterns. Depictions of the remaining clusters can be found in the supplementary material. Cluster sizes are variable and the mean size of clusters for each state can be seen in table 3.2.1.

The cluster found within the state of Rio de Janeiro can be seen at figure 3.7. The clusters for Ceará and Paraná are in the Appendix chapter.

Figure 3.8 show the clusters in the map for each state. Note that geographical proximity is not the only predictor of cluster membership.

3.2.2 Forecasting

We measured the performance of the forecasting models by the magnitude of the prediction error for week $t + 4$ where t is the last week observed. The prediction errors were calculated as mean squared error (MSE), mean absolute error (MAE) and mean absolute scaled error (MASE).

In order to measure the gain value of the clustered dataset (described in section 3.1.2) we also trained the lasso and random forest models without the features series from its clusters, having as input to the model only the time series of the city we are forecasting. Than, we compared the losses of

both forecasting to choose the best data modeling.

In figures 3.9, 3.11 and 3.10 we compared the models losses out-of-sample for each type of input data (clustered and not clustered). We can see that, in general, the models with cluster data have less error variance than the single models, so we used for the next forecasting results the models trained with the features series from the clustered data.

The comparison between the models efficiency is shown in Figure 3.12 for each state separately. The LSTM models returned minor error than the lasso and Random Forest when looking at the MASE and MSE losses for all the states. It is interesting to notice that the lasso model have performed better than the Random Forest for the state of Rio de Janeiro, but in Paraná and Ceará it had lower performance than the Random Forest.

We calculated a QQplot (Figure 3.13) for each state with the goal to understanding the predicted value belongs to the empirical distribution of the data. The empirical distribution used to construct the QQplot, was calculated by approximating a distribution to the historical incidence of each week through the years; for example, we are comparing if the prediction value for the first week of the year is close to the values of this same week in the historical data. We can see that the LSTM has achieved a higher proportion of predictions within the historical distribution.

The figure 3.14 shows, the relation between the loss function MASE for each city within a state and the state population size and the state total incidence.

Although the relationship between the MASE and the population size does not show a clear correlation, the MASE x total cases plot show a proportional increasing relationship (more easily to see at the Paraná plots, since it has more cities). We believe that this happens because in cities with a high incidence, we have a bigger variance between the incidence values from epidemic weeks and non epidemic weeks. This makes the models to have difficulty to predict at those extremes, increasing the errors in those cities.

We also have the boxplot of the the losses for each model. The boxplot shows us that in general the LSTM achieved smaller errors.

Figures 3.15, 3.17, and 3.16 show the performance of the prediction both *in-sample* and *out-of-sample*, for some cities in the states of Rio de Janeiro, Paraná and Ceará. The full cities predictions are in the appendix chapter.

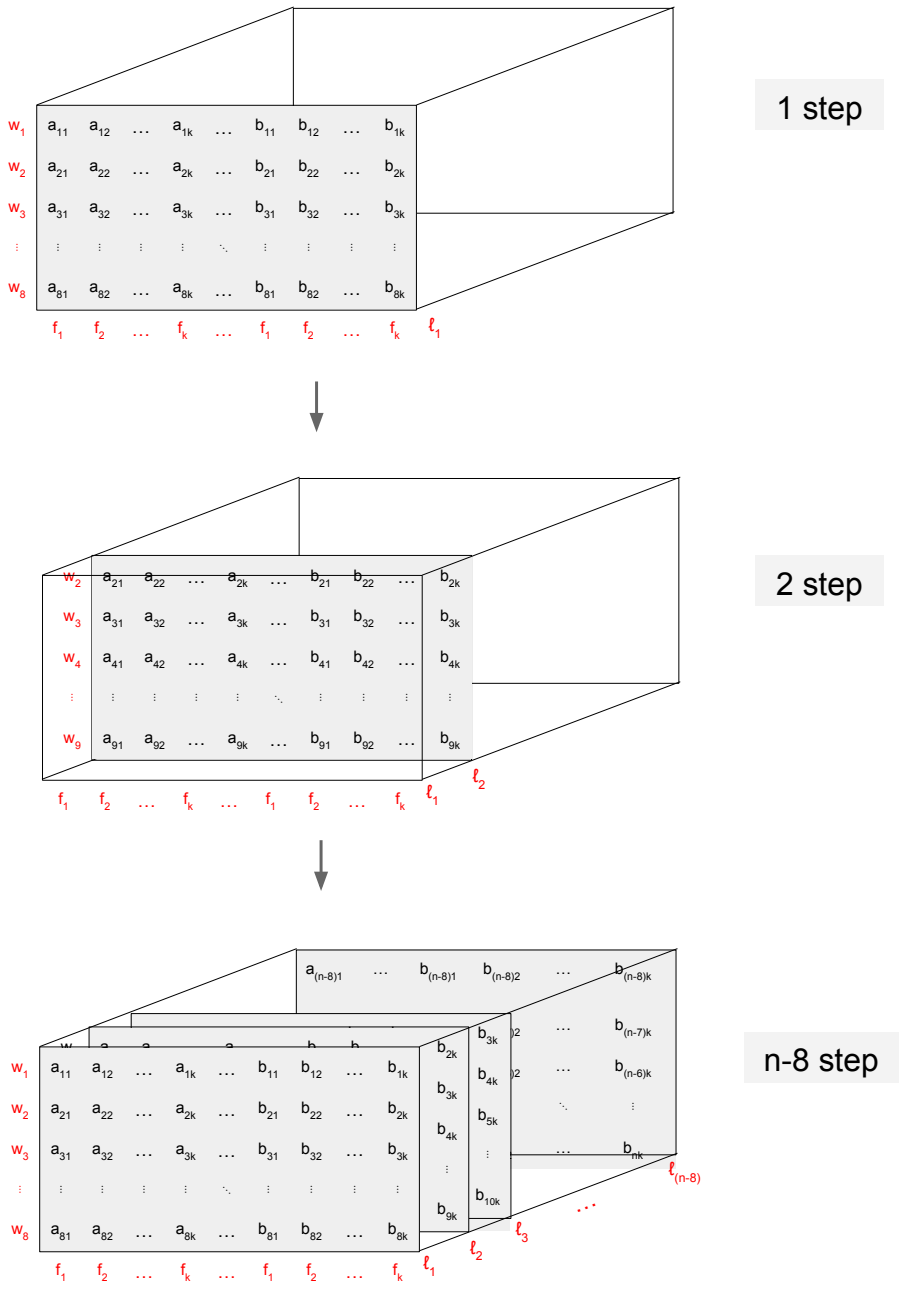

Figure 3.5: Creation of features matrix (tensors) used to train the LSTM model.

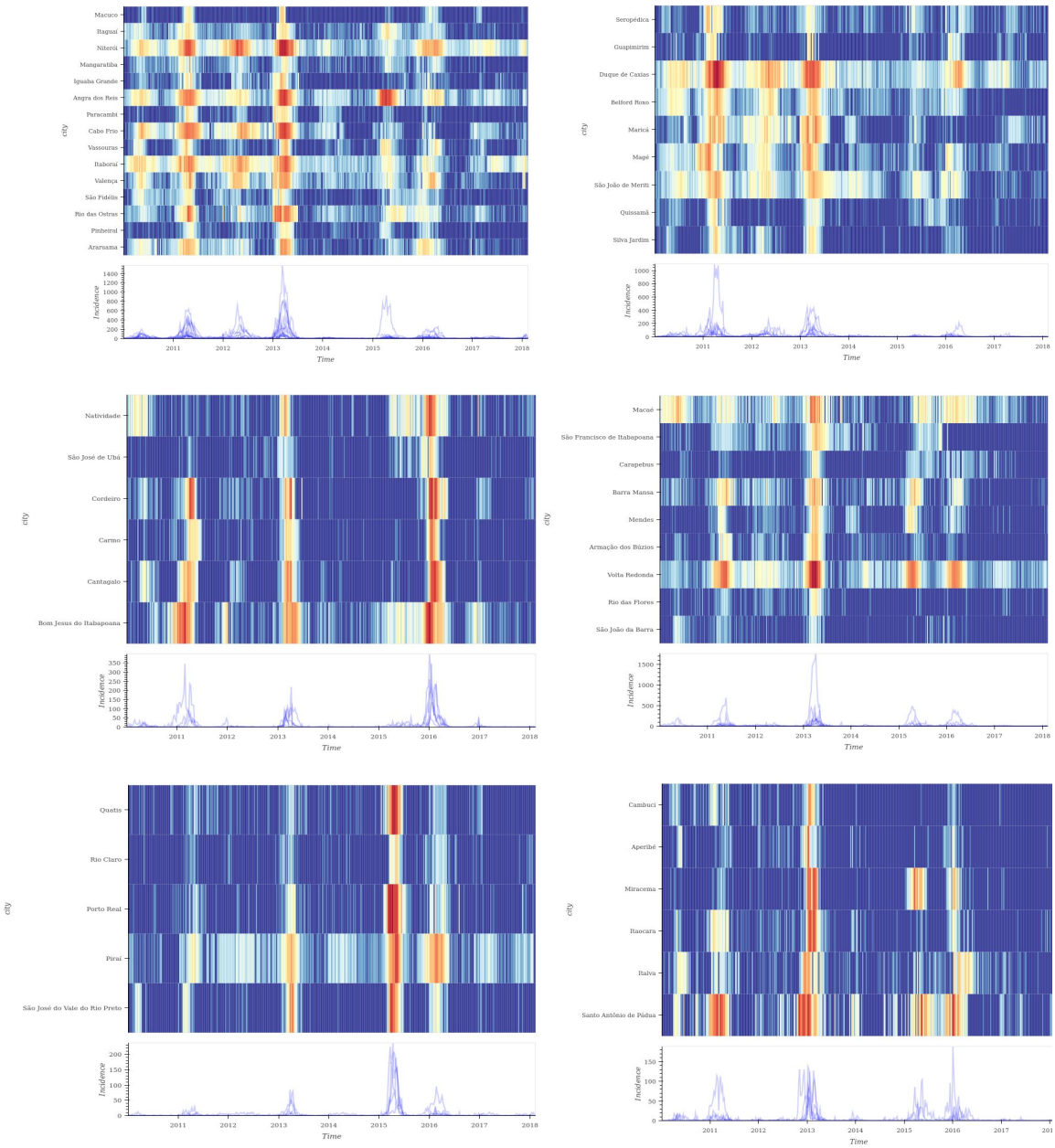

Figure 3.6: Matrix colormap of dengue incidence for some clusters from Rio de Janeiro state. The colors represent the incidence of dengue through time, with red being the highest incidence and the dark blue the lowest.

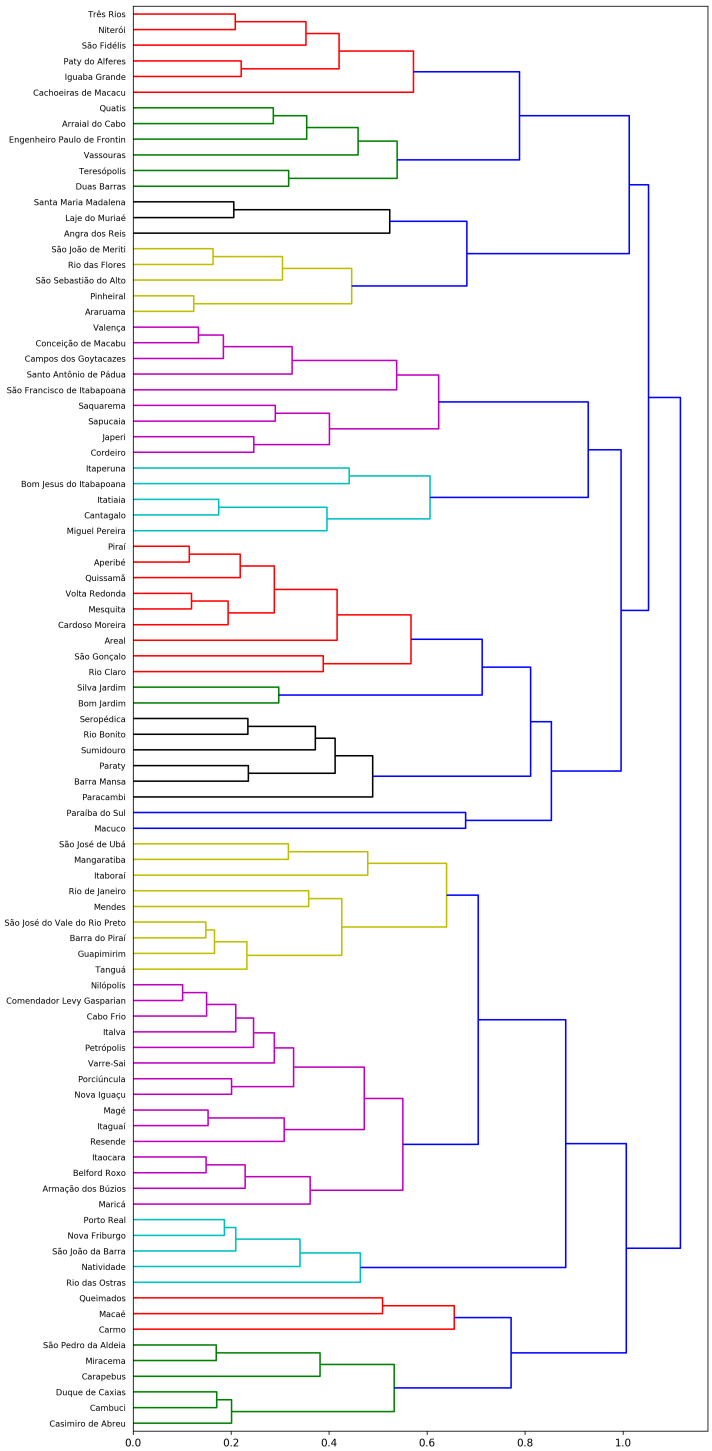

Figure 3.7: Hierarchical cluster of Rio de Janeiro.

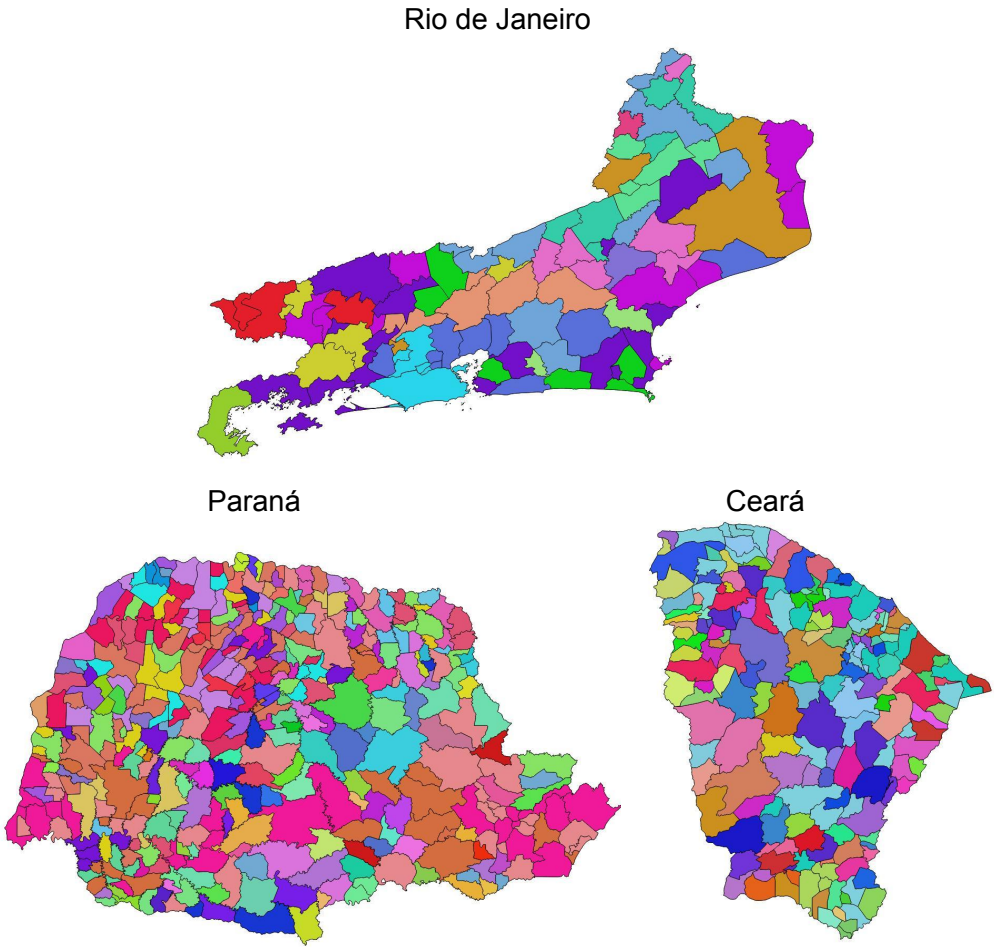

Figure 3.8: Map of Hierarchical clusters for each state.

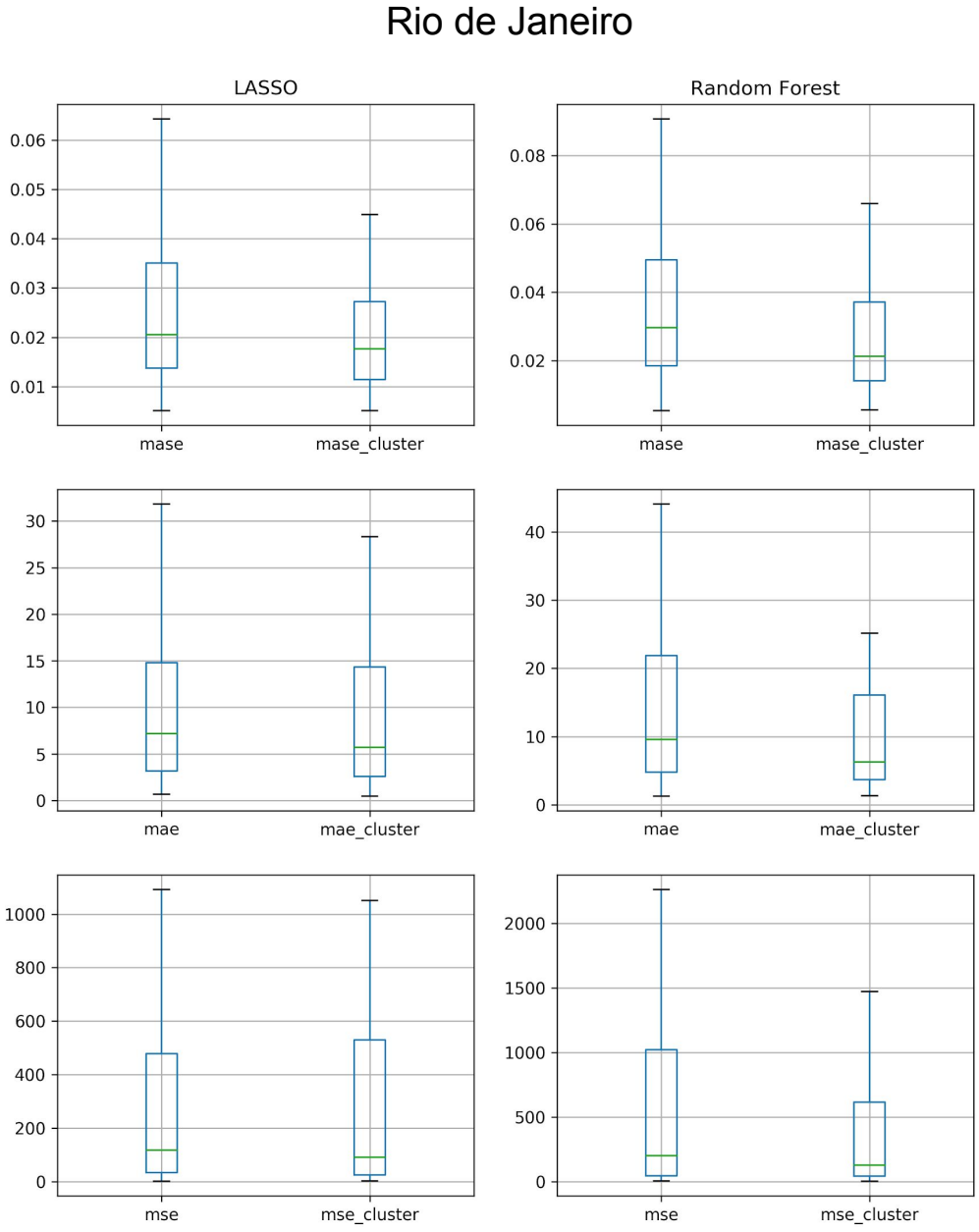

Figure 3.9: Loss comparison between models trained with and without cluster features for Rio de Janeiro state.

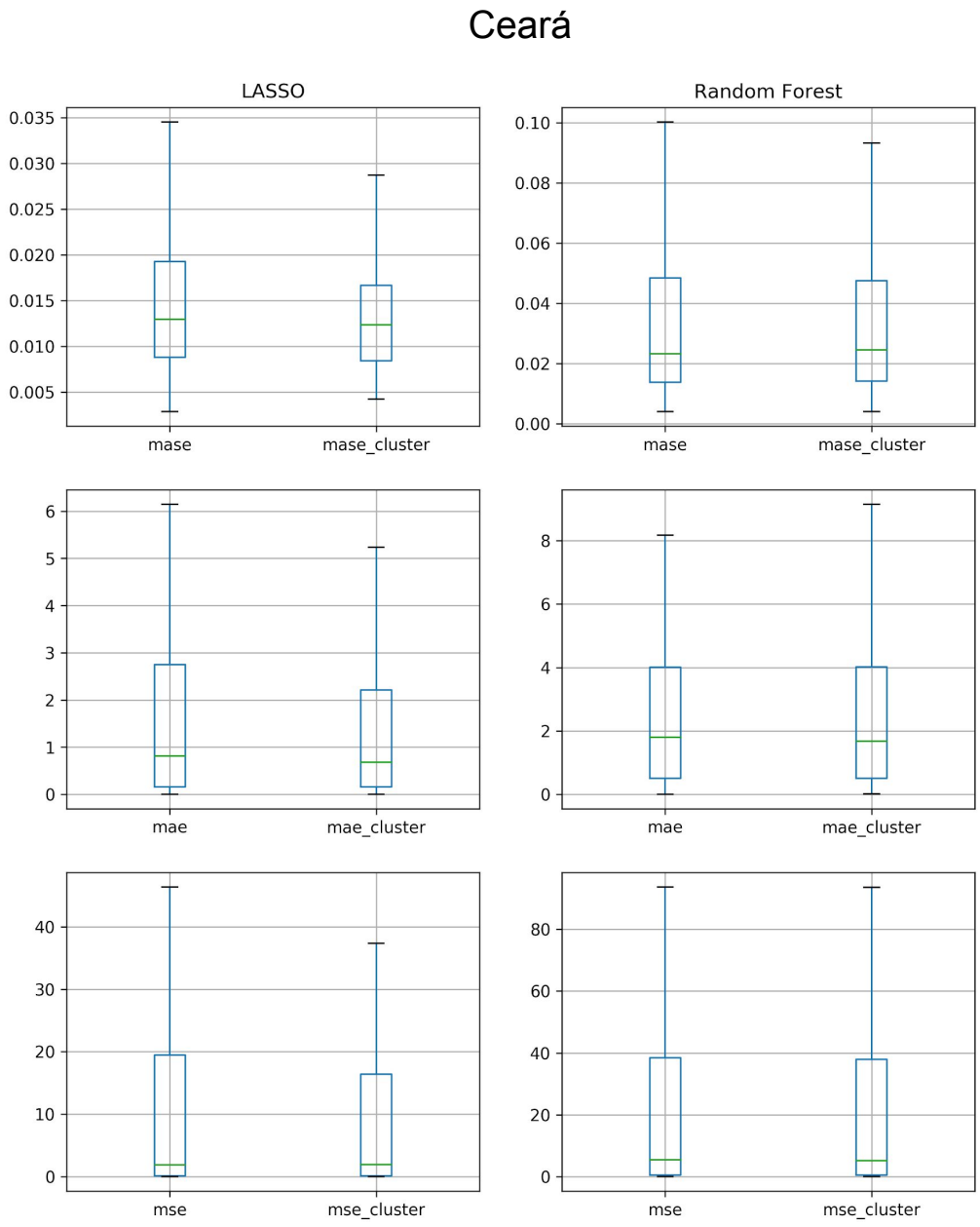

Figure 3.10: Loss comparison between models trained with and without cluster features for Ceará state.

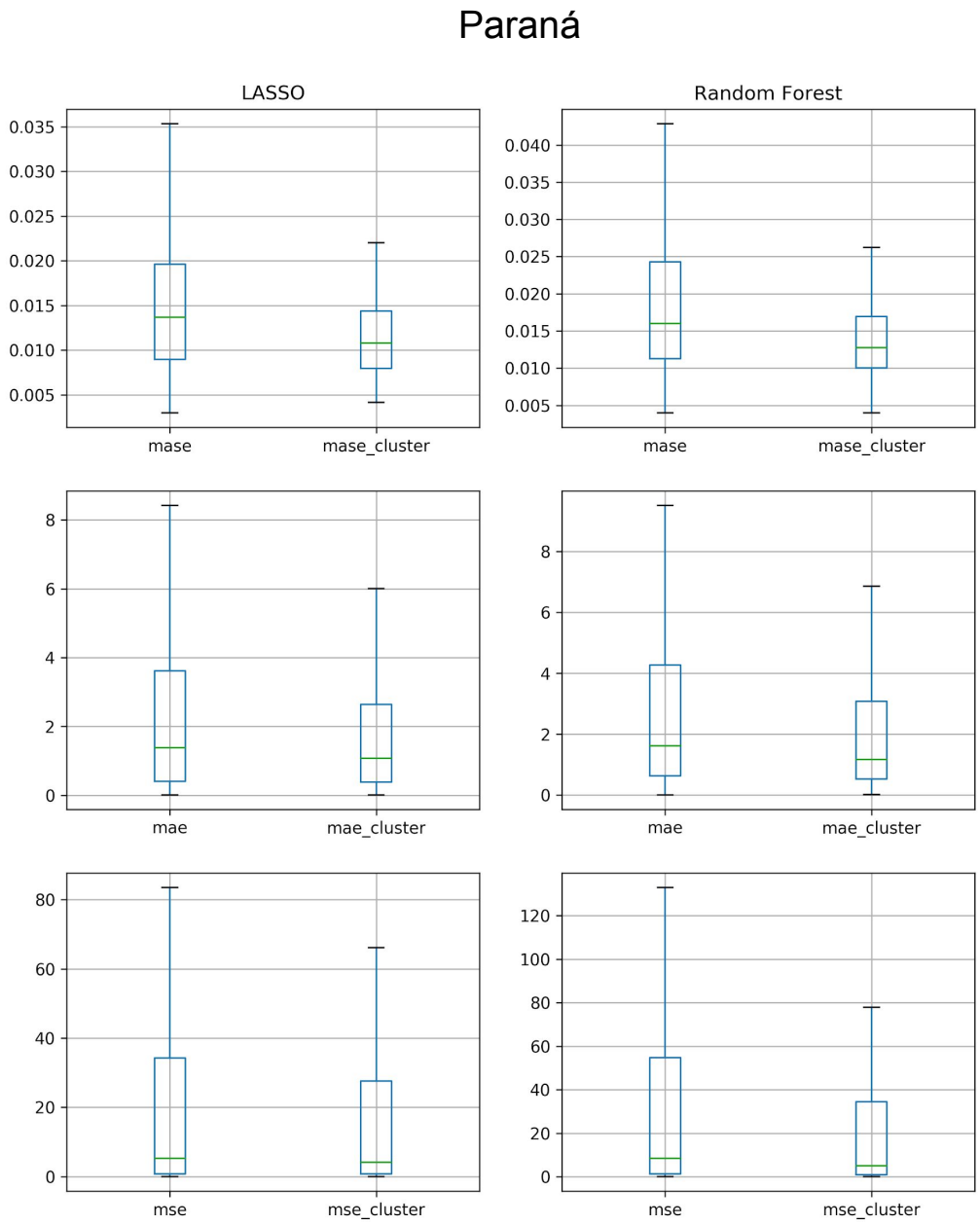

Figure 3.11: Loss comparison between models trained with and without cluster features for Paraná state.

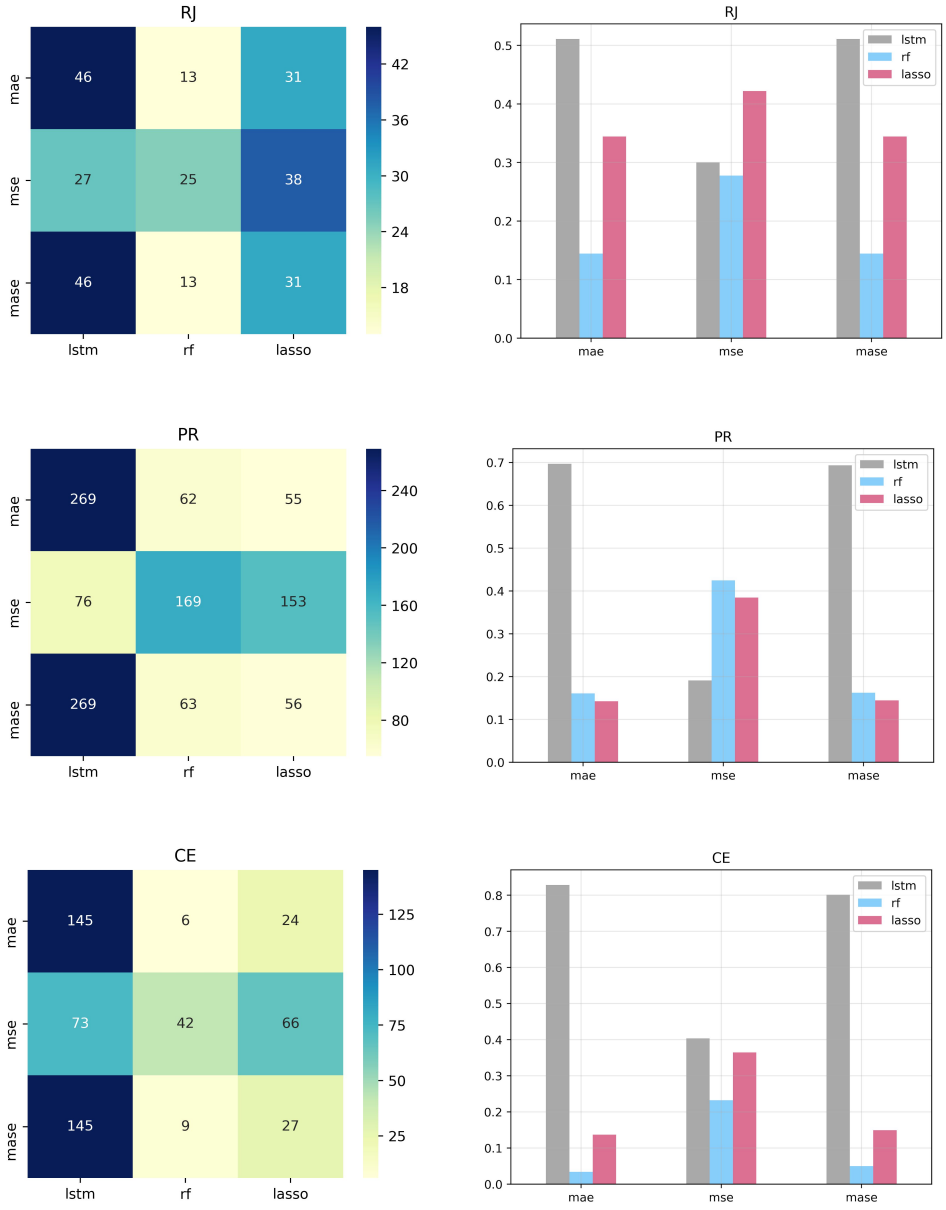

Figure 3.12: Count of errors for each model. The left column contains the heatmap matrix for each state with the count of times that a specific model performed better for each type of loss. The right column contains the percentage bar plot where of 'winning' performance for each loss.

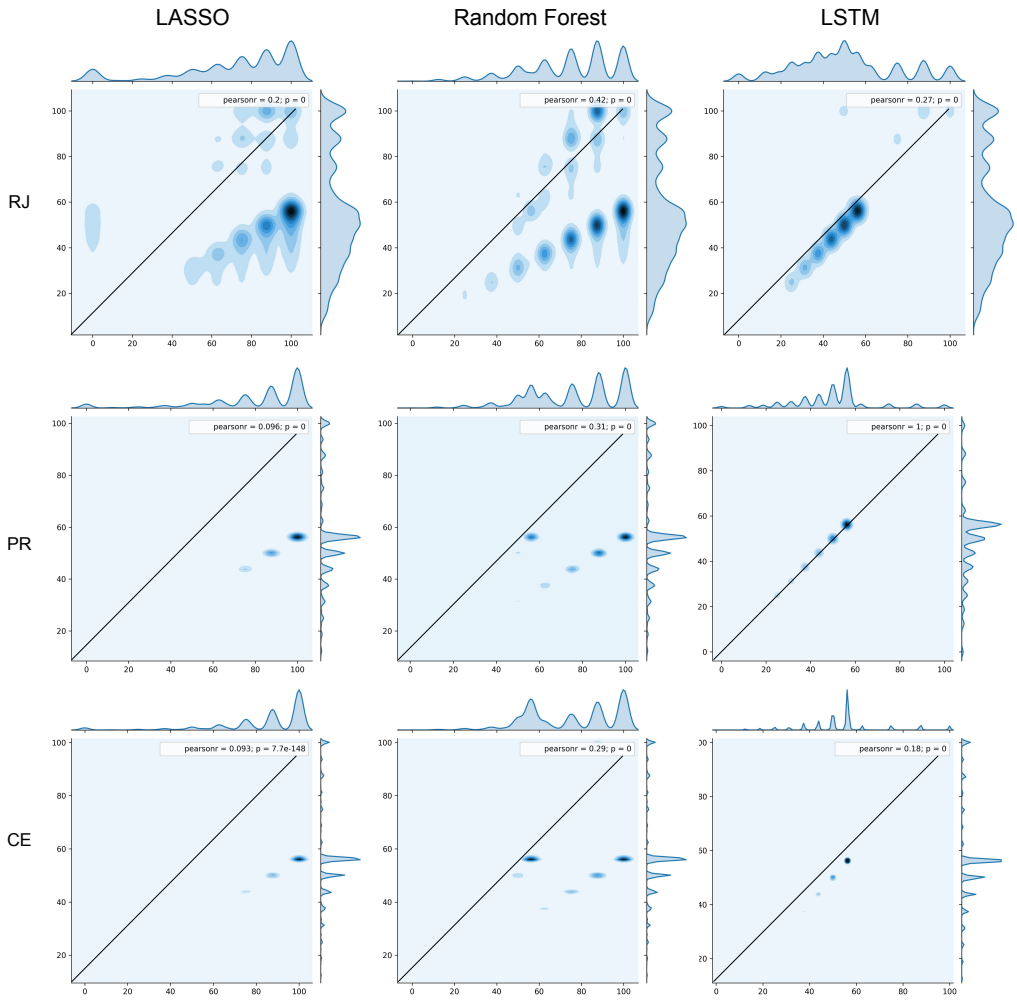

Figure 3.13: Each pair of points (x,y) is the empirical distribution percentile where the predicted and real value belong, respectively, for each city of the state. If the predict value and the real value of a week i is near the line $x = y$, than the predict value is consistent with the historical data. We plotted the density of the points, so the dark blue area is the area where the most values belongs to.

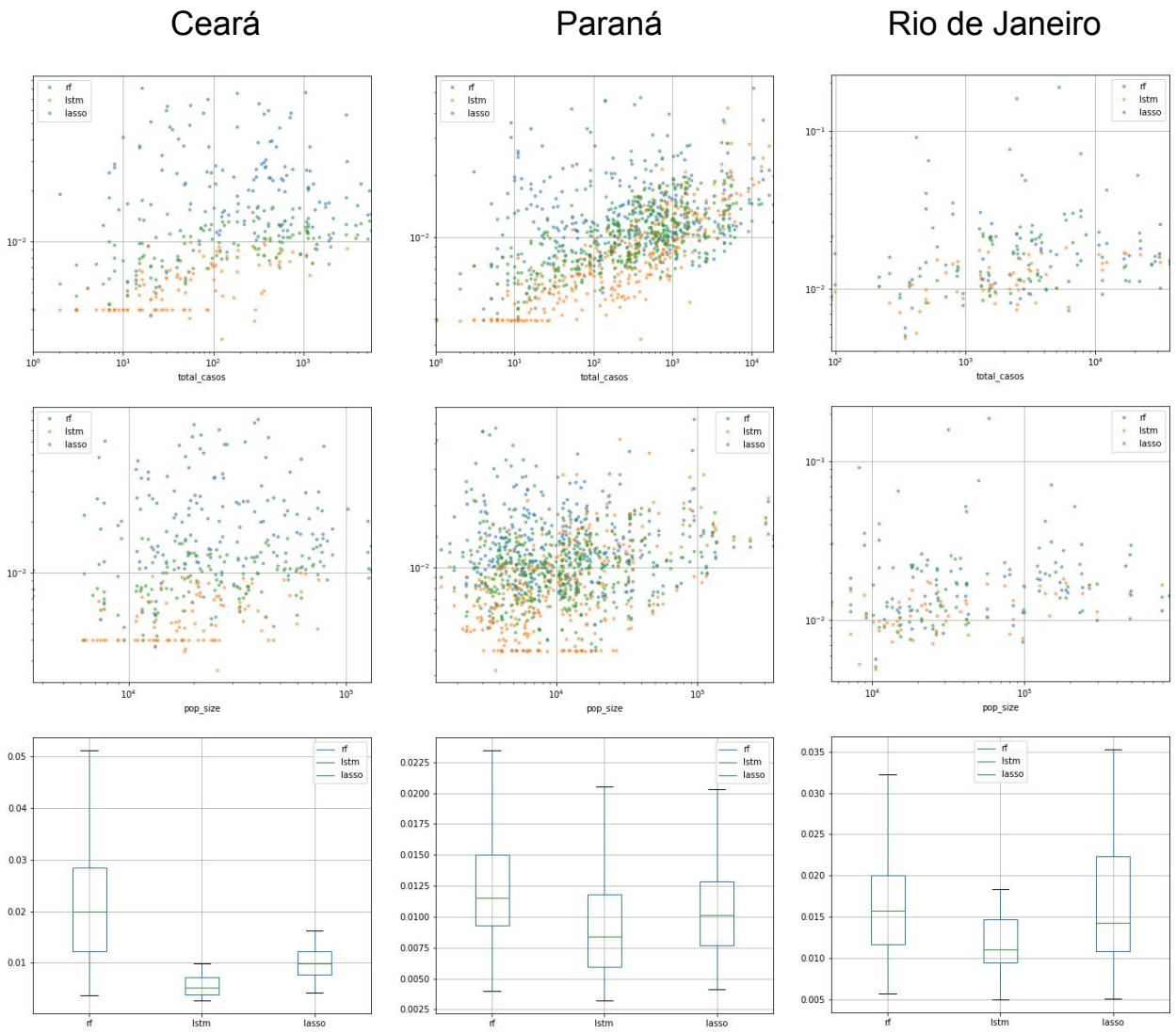

Figure 3.14: MASE loss function analysis. The first two rows are scatter plots of the MASE loss value for each city within a state. The first row is the relationship with the total number of dengue cases for those cities, and the second row is the relationship with the population size. In the last row we have the loss boxplot of the loss function for each trained model.

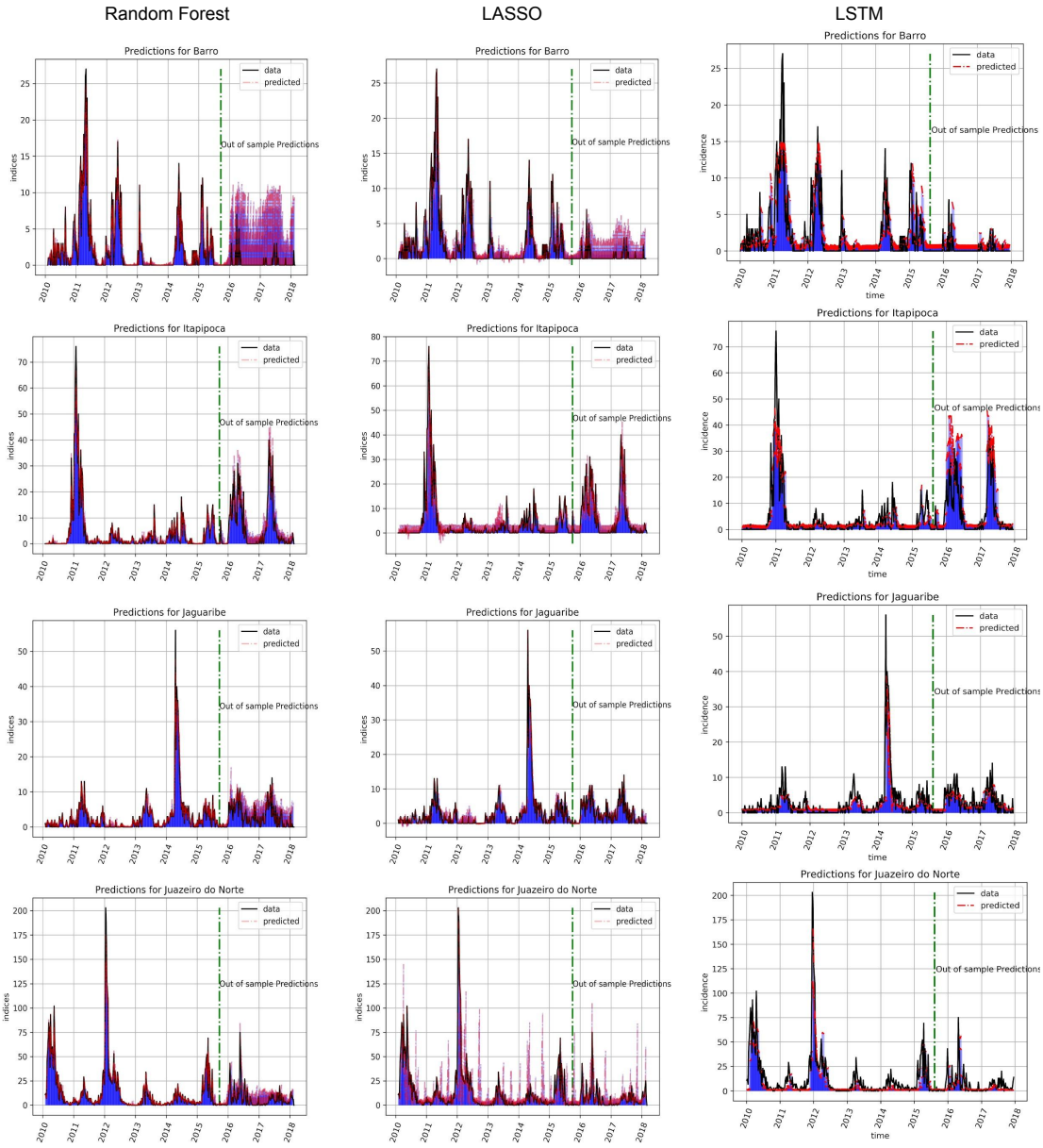

Figure 3.15: Forecasting for cities of Ceará. The red lines are the predicted value for each τ of the prediction window. The green line delimit the train set (in-sample) and the test set (out-of-sample)

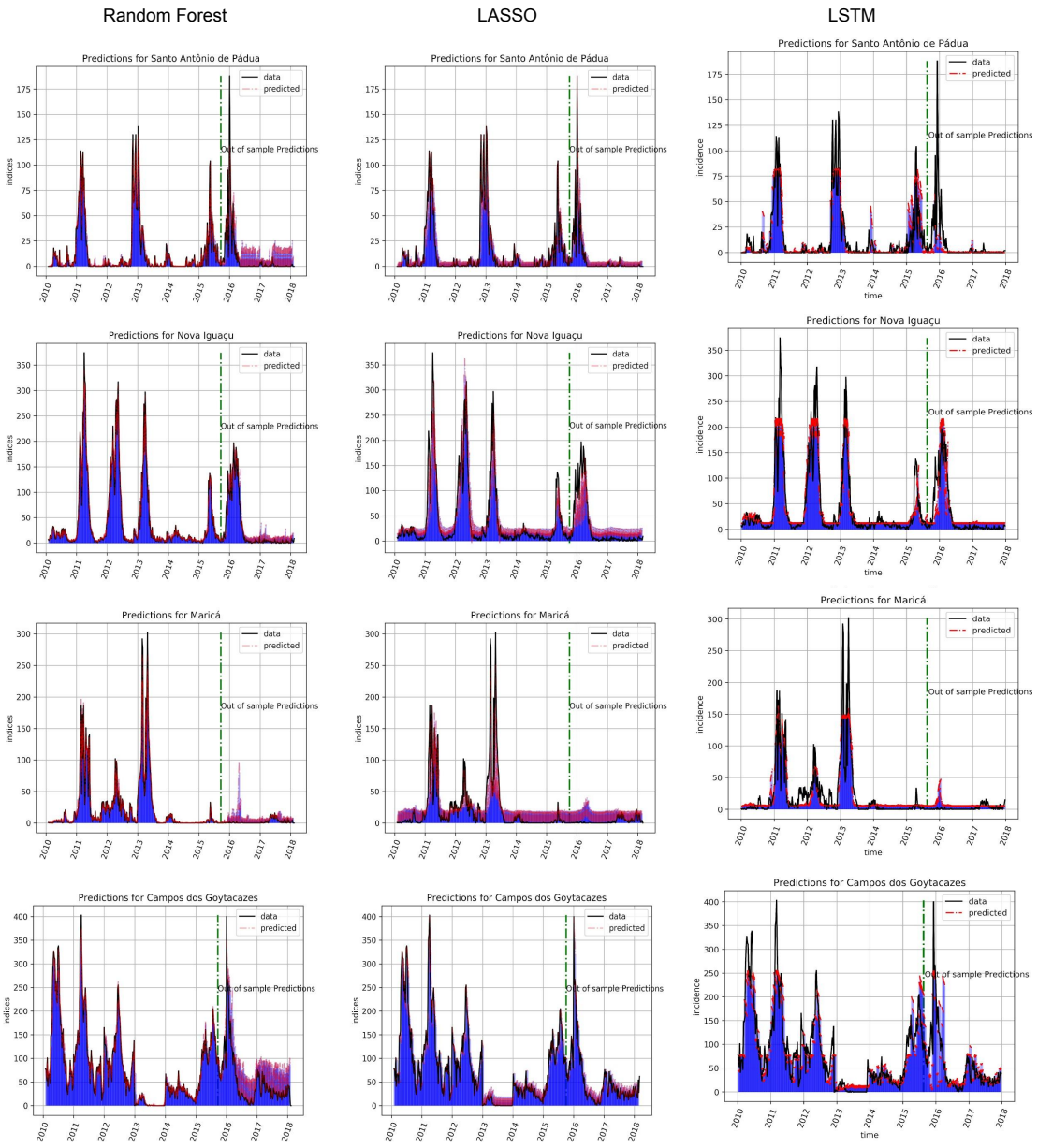

Figure 3.16: Forecasting for cities of Rio de Janeiro. The red lines are the predicted value for each τ of the prediction window.The green line delimit the train set (in-sample) and the test set (out-of-sample)

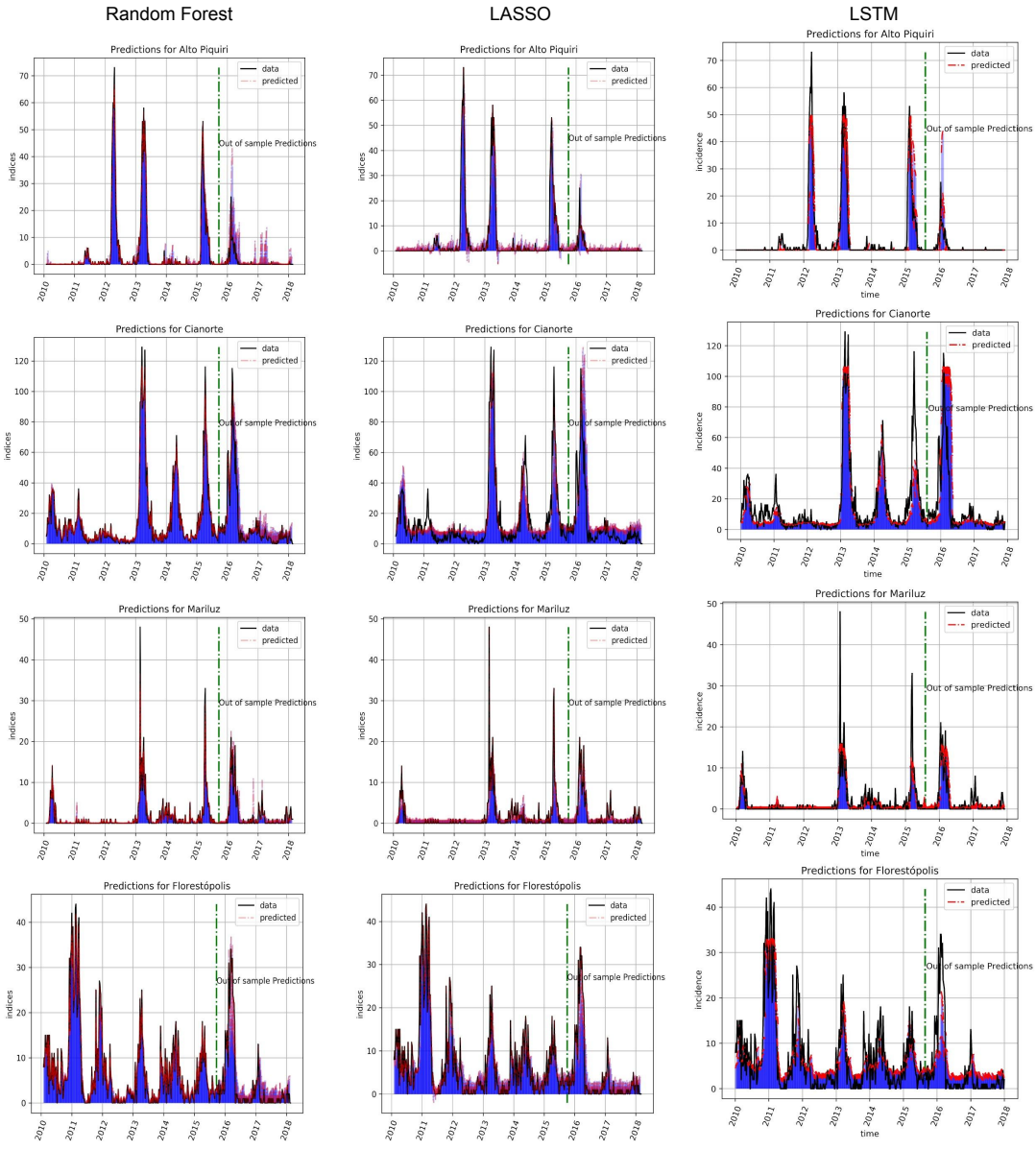

Figure 3.17: Forecasting for cities of Paraná. The red lines are the predicted value for each τ of the prediction window. The green line delimit the train set (in-sample) and the test set (out-of-sample)

Chapter 4

Conclusions and Final Considerations

All three models have shown reasonably good performance for both large and small cities from various parts of Brasil. The LSTM model, however displayed the best overall performance, meaning that on average, it displayed smaller prediction errors.

When the models were trained using clusters of cities as predictors, the results were consistently superior than when using just local predictors. This shows that the clustering methodology proposed enhanced the abilities of the models to learn the epidemic dynamics. The extra information provided by the sister cities' series allowed the model to substantially outperform the base model.

Another important feature of the LSTM model, was that it was capable

of consistently predict the incidence pattern of non-epidemic years, leading to very few false positive predictions of outbreaks.

One weakness of the LSTM model was that it didn't predict well the magnitude of the peak of epidemics, but it showed consistency when predicting the start of the epidemic, which is the most valuable information when forecasting epidemics.

We believe that the this difficulty to match high values in the series is linked to the use of a mean square log error (MSLE) loss function for the training of the neural networks. This loss function gives less relative weight to higher values due to the logarithm. On the other hand, as the LSTM has the tendency of underestimate the epidemics which is good when thinking of higher costs of false positives for the already strained state public health resources.

Random Forest and lasso models showed good results but when comparing out of sample predictions, they were not as consistent as the LSTM model.

The models performed well in some cities and poorly at others. It's interesting to analyze each city and learn which model performs well on each case. We believe it would be worth building an ensemble of the three models for optimal predictive power in a production setting, but we have not gone as far as doing that in this work.

Machine and deep learning are rapidly growing areas with a lot of potential in forecasting. The LSTM model has a lot of space for improvements,

once the gates of the cells can be changed to satisfy the specificity of the problem, and achieve better results, as shown in Cho et al. (2014), Koutnik et al. (2014), Yao et al. (2015) and Xia et al. (2009) . We believe that a improvement can be made to satisfy epidemics forecasting in future works.

It's important to remember that LSTM has a much higher computational cost to train which can make random forest and lasso better options in situations of limited computational resources.

Bibliography

P. M. Luz, B. V. Mendes, C. T. Codeço, C. J. Struchiner, A. P. Galvani, Time series analysis of dengue incidence in rio de janeiro, brazil, The American journal of tropical medicine and hygiene 79 (2008) 933–939.

S. T. Stoddard, B. M. Forshey, A. C. Morrison, V. A. Paz-Soldan, G. M. Vazquez-Prokopec, H. Astete, R. C. Reiner, S. Vilcarromero, J. P. Elder, E. S. Halsey, et al., House-to-house human movement drives dengue virus transmission, Proceedings of the National Academy of Sciences 110 (2013) 994–999.

C. Codeco, O. Cruz, T. I. Riback, C. M. Degener, M. F. Gomes, D. Villela, L. Bastos, S. Camargo, V. Saraceni, M. C. F. Lemos, et al., Infodengue: a nowcasting system for the surveillance of dengue fever transmission, BioRxiv (2016) 046193.

G. E. Box, G. M. Jenkins, G. C. Reinsel, G. M. Ljung, Time series analysis: forecasting and control, John Wiley & Sons, 2015.

H. Tong, Non-linear time series: a dynamical system approach, Oxford University Press, 1990.

L. L. Scharf, C. Demeure, Statistical signal processing: detection, estimation, and time series analysis, volume 63, Addison-Wesley Reading, MA, 1991.

M. R. Anderberg, Cluster analysis for applications, Technical Report, Office of the Assistant for Study Support Kirtland AFB N MEX, 1973.

V. Estivill-Castro, Why so many clustering algorithms: a position paper, ACM SIGKDD explorations newsletter 4 (2002) 65–75.

L. Rokach, O. Maimon, Clustering methods, in: Data mining and knowledge discovery handbook, Springer, 2005, pp. 321–352.

S. Shen, H. Jiang, T. Zhang, Stock market forecasting using machine learning algorithms, Department of Electrical Engineering, Stanford University, Stanford, CA (2012) 1–5.

M. Santillana, A. T. Nguyen, M. Dredze, M. J. Paul, E. O. Nsoesie, J. S. Brownstein, Combining search, social media, and traditional data sources to improve influenza surveillance, PLoS computational biology 11 (2015) e1004513.

Y. Bai, Z. Jin, Prediction of sars epidemic by bp neural networks with online prediction strategy, Chaos, Solitons & Fractals 26 (2005) 559–569.

L. Breiman, Random forests, Machine Learning 45 (2001) 5–32.

C. Olah, Understanding lstm networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015.

Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE transactions on neural networks 5 (1994) 157–166.

S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.

L. Eisen, S. Lozano-Fuentes, Use of mapping and spatial and space-time modeling approaches in operational control of aedes aegypti and dengue, PLoS Negl Trop Dis 3 (2009) e411.

B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, et al., Least angle regression, The Annals of statistics 32 (2004) 407–499.

R. S. Olson, N. Bartley, R. J. Urbanowicz, J. H. Moore, Evaluation of a tree-based pipeline optimization tool for automating data science, in: Proceedings of the Genetic and Evolutionary Computation Conference 2016, ACM, pp. 485–492.

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078 (2014).

J. Koutnik, K. Greff, F. Gomez, J. Schmidhuber, A clockwork rnn, arXiv preprint arXiv:1402.3511 (2014).

K. Yao, T. Cohn, K. Vylomova, K. Duh, C. Dyer, Depth-gated recurrent neural networks, arXiv preprint (2015).

J. Xia, F. Chen, J. Li, N. Tao, Measurement of the quantum capacitance of graphene, Nature nanotechnology 4 (2009) 505.