A Machine Learning approach to Dengue Weekly Cases Regression problem

Vincenzo Marciano'
Politecnico di Torino
Student id: s282004
s282004@studenti.polito.it

***Abstract*—In this study a suitable solution to predict weekly cases for Dengue Fever spreading in two locations is proposed. Since the task seems to be a possible *forecasting* problem, several temporal data analysis and transformation are suggested to create a recognizible pattern for the prediction. As a following action, different learning models are trained on processed data, choosing the most valuable option in the hyperparameters settings. Eventually, the tuning process will detect the best setup for the best model, which would achieve an higher performance than the suggested baseline.**

I. Problem overview

First and foremost, the dataset for the proposed task was already divided into two sets, which may be considered for training and testing purposes respectively:

- *development set* (1205 instances), a *tsv* file which contains all the relevant features that could be adopted for the problem analysis, along with the target values (***'total_cases'***);
- *evaluation set* (364 instances), another *tsv* file that has some samples described with the same features showed in the development set to be used for the testing phase as new records for the model. In addition to that, this particular dataset won't be used as a whole, but subdivided into *public* and *private* set, with the former eventually exploited for the experiments.

The target labels are positive numerical values that represent the **weekly number of cases** of Dengue fever, spreaded thoughout two different locations, *Iquitos (Peru')* and *San Juan (Puerto Rico)*. The categorical attribute *city* recognizes these two cities as **iq** and **sj** respectively. Also, some temporal informations are retrieved by the *year*, *weekofyear* and *week_start_date* columns. The weekly atmospherical background of each city is then described by various numerical features, given by satellite measurements.

Thanks to some exploratory data analysis, it can be seen that the original development set beneficies from the absence of **duplicate** objects. However, many **missing values** are recorded: at a first glance, they are either concentrated within a certain subset (*avg_temp_c, max_temp_c, min_temp_c, diur_temp_rng_c, precip_mm*) or the entire set of atmospherical attributes. Interestingly, the latter situation is referred only to the first day (*01/01/....*) for some years. Even if NaN values represent a small fraction of the whole training set, discarding them may be not the right solution: as an example, at a certain point we may have to deal with five continuative rows that have missing information along the aforementioned subset of features. This situation is quite critical for the model, because we are dealing with more than a month of missing data.

Another noticeable assuption concerns the location: despite the imbalanceness of the development set between the two cities' division, evaluation data is fill with istances belonging to Iquitos. Hence, a first idea might be to ignore the San Juan samples, to build a more Iquitos-oriented model. In addition to that, Iquitos training set observes an initial sequence of null cases that could not be so useful for the proposed regression architectures.

II. Proposed approach

To address some of the issues presented in Section I, a preheventive study has to be done. According to some researches, the previous "zero-cases" assumption for Iquitos is confirmed [1]: Peruvian Dengue fever began to spread from late 2001, for this reason we can ignore the related samples without any concern, since they bring no useful details in the evolution of the pandemic. Further evidence can be retrieved in Figure 1 .

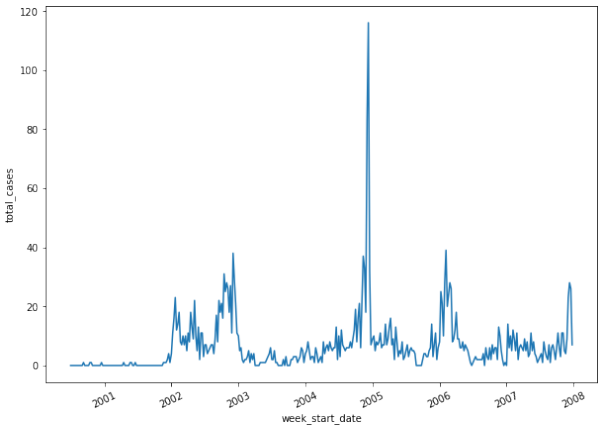

Fig. 1. Total weekly cases spread along the years in Iquitos, Peru'

Another important contribution could be given by the missing values manipulation. Therefore, as a first step we can think to set a threshold for the number of missing values of each row, to delete those rows that are filled with more than 5 NaNs.

A. Preprocessing

Dropping instances with a huge presence of missing data is the starting point to mitigate this problem. Consequently, we have to find the proper tool to manage the "temporal holes" created by the remaining NaNs, as it was stated in the previous Section. **KNNImputer** [2] is selected as the best choice in assessing this task: it imputes the missing value by exploiting a k-Nearest-Neighbors approach, i.e. computing the mean value from *n* nearest neighbors found in the training set, defining two samples "close" if the features that neither is missing are close. For our convenience, the imputer's parameter *n_neighbors* is set to 5, being careful to apply the *transform* method on the test set once it was fitted on the training one. An alternative way could be the **Simple Imputer** [2], but after various experiments it confirms its bad generilization in retrieving the surrounding samples' behaviour of the missing value imputed, with respect to the used approach.

Following the common pipeline, a further step in the preprocess is to undergo through various techniques for the dataset dimensionality reduction. Firstly, the following attributes are removed since they are poorly relevant:

- *week_start_date* brings no further informations. A first attempt in achieving other temporal data from this attributes has been conducted (month, quarter, national holidays), but they seems to be useless for the built regressors.
- *year* is not strictly related to the periodical evolution of the pandemic: instead we might focus on *weekofyear* by looking at the seasonal evolution of the disease, and using the index as a time forward counter.

Among those, a **correlation matrix** could help in selecting and avoiding some features which may be very high correlated between each other. The threshold of "high-correlation" is not known a priori and it may vary in a case-by-case scenario: for our experiment, the columns dropped experienced a *Pearson* correlation greater than **0.85** in absolute value, as a result of the following experession:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

$$r \in [-1, +1]$$

The resulting features achieve low correlation score (in absolute value) with respect to *total_cases*, as it's shown in Figure 2. However, this is not to be considered as a negative factor: Pearson correlation spreads along the linear plane and for this reason some features might be highly correlated with the target in other spatial projections [3].

A further study is conducted within the possible presence of **outliers**. In particular we focus on the overall total cases and on the attributes referred to *temperature*. For what concerns the total cases evaluation, Figure 1 suggests an anomalous value peaking up the curve to 116 cases registered in one week: we may suppose this happened by registering previous forgotten cases. Then, Figure 3 displays the behaviour of the three atmospherical features; also here we experience some

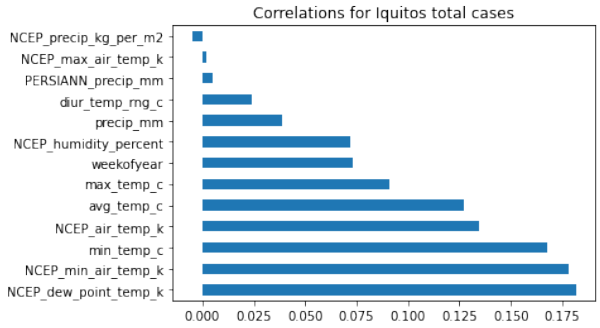

Fig. 2. Correlation Score for each feature with respect to the target.

irregular records: for example, one of them is the maximum temperature registrated in the summer of 2003 (greater than 40 °C), which is quite inconsistent with the overall behaviour and may be misleading for the regressors analysis. A solution to manage these peculiar values is to remove them from the final dataset.

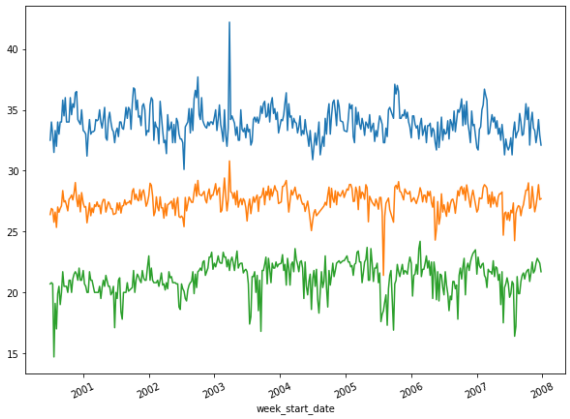

Fig. 3. Maximum(blue), Average(orange) and Minimum(green) weekly temperature expressed in °C and measured in Iquitos, Peru

Lastly, each of the exploited model may be feeded by different input data, so **scaling** may be taken into account: Random Forest and Gradient Boosting regressors only needs consistent numerical values, without neither normalization or possible one-hot-encoding for categorical values (since we already operate on the Iquitos subset); on the other hand Support Vector regressors may need a scaled form of the training data: for our convenience, a **Standard Scaler** is adopted.

The used datasets are then summarized as it follows:

A All features are maintained;
B Subset of features;
C Outliers removed from B.

B. Model selection

We may now focus on the used Regression models of our task:

- **Random Forest** regressor relies on a tree-based ensemble idea. Here the dataset is bootstrapped into *B* different

subsets with resampling, then for each of them a fixed number $m < p$ features is selected. Eventually, from each subset a decision tree is trained and the overall prediction is obtained averaging each individual predictor response [3].

- **Support Vector** regressor is based on the optimal hyperplane separator and margin maximization, i.e. it acts on the same principles of SVM Classifiers. Given a p-dimensional space, an hyperplane is a (p-1)-dimensional subspace described by a certain equation: SVMs use this separator to discriminate the various values in the entire set, along with an evaluation interval for each instance (the margin). As well as the need of selecting an ϵ_i quantity to evaluate the constraints which allow some values to be within the margin or wrong halfspace, SVR enjoys the presence of a *Kernel* function that allows to perform a separation even if it's done within non-linear spaces [3].
- **Gradient Boosting** regressors follows the general idea behind Random Forest, that is a tree-based ensemble method in which each bootstrapped subsets of training data gets a subset of the original feature set. However the main difference relies in the costruction of every predictor: each tree is constructed by looking to the previous iteration, creating a *reweighted* version of the training set [3]. Between all the libraries and novel implementations that surround GBM, one of them caught our interest for the experiments: **eXtreme Gradient Boosting** (XGB). This is considered one of the most popular Machine Learning algorithm to deal with. Since its release, XGB became the state-of-the-art approach for various structured data: among a vaste group of regression or classification problems, it actually excels with time series data and for this reason it may be the best fit for our regression task. However, one of the crucial advantages of XGB is the *second order* gradient computation (Newton's Method): the loss function is now minimized by looking at its second order partial derivatives, that retrieve more informations with respect to the base model [4].

C. Hyperparameters tuning

For the hyperparameters tuning a *GridSearch* [2] strategy with K-Fold Cross Validation is adopted, where the number of folds is set to 3. For each model a proper search space of paramaters is provided and summarized in Table I, II and III

TABLE I
HYPERPARAMETERS SET FOR RANDOM FOREST

Parameters	Values
n_estimators	250,500,1000
criterion	mae
max_features	auto,sqrt,log2
max_depth	3,5,7
min_samples_split	1,5,10
random_state	42
n_jobs	-1

TABLE II
HYPERPARAMETERS SET FOR SVM

Parameters	Values
kernel	rbf,poly,linear
C	1, 10, 0.1, 100
gamma	1e-4, 1e-3, 1e-2, 0.1, 1
epsilon	0.1, 0.5, 0.01

TABLE III
HYPERPARAMETERS SET FOR XGB

Parameters	Values
n_estimators	250,500,1000
learning$_rate$	0.1,0.001,0.01
subsample	0.3, 0.7, 1
colsample_bytree	0.5, 0.8
max_depth	3, 5, 7
random_state	42

Even if Random Forest and XGB beneficies of a moltitude of hyperparameters to be tuned, this results in a considerable issue: the outcomes may vary case-by-case and the research among all the folds for the best configuration may be computationally infeasible. For this reason a pruned set of parameters must be imperative, yet adequate.

The scoring function for our search to work with is the **Mean Absolute Error**, retrieved by the absolute value of the Negative MAE (a built-in evaluation scoring of the grid search method). It's expressed as:

$$\text{MAE} = \left(\frac{1}{n}\right)\sum_{i=1}^{n}|y_i - x_i|$$

Where each y_i and x_i represent the predicted and true values respectively. The regression goal is to minimize this function: in other words, the lower, the better.

III. RESULTS

The final step is to compare the outcomes achieved with the aforementioned combination of models and hyperparameters, for each proposed training set.

TABLE IV
LOCAL MAE BY EACH MODEL FOR EACH TRAINING SET

Training Set	RF, XGB, SVR		
Training Test A	5.87	5.80	5.71
Training Test B	5.89	5.79	5.44
Training Test C	5.56	5.50	5.14

With a ongoing trend of performances, Table IV suggests as a good strategy in decreasing MAE the consecutive removals of high correlated features and then outliers. Intuitively, for the submission we have to consider working only with Training Set **C**.

Apparently, the SVM Regressor is the one which locally performs the best. However, in the public evaluation set, **XGB** confirms its outstanding work obtaining a MAE of **5.618**,

even if a few hyperparameters were fine-tuned. XGB also outperforms Random Forest and SVR, whose scores are 5.67 and 5.76 respectively. The hyperparameters choices for the best model are then summarized in the Table V.

TABLE V
BEST POOL OF HYPERPARAMETERS FOR XGB

Parameters	Values
n_estimators	1000
learning_rate	0.001
subsample	0.7
colsample_bytree	0.5
max_depth	3
random_state	42

IV. DISCUSSION

The achieved results reflect the Gradient Boosting generalization capability and the reason why XGB is actually the most used in recent competitions. Also, the 3-folds Cross Validation expressed in the GridSearch helps in avoiding overfitting occurrence. Eventually, all three models behave very well, maintaining their scores below the 6-value threshold for MAE.

A further analysis might be conducted in evaluating the *SanJuan* behaviour, searching for both atmospherical and weekly trends. In this case, a **CatBoost** [5] regressor maybe a right choice, since its re-introduce the categorical attribute related to the location. In addition to that, one could focus on the reproduction cycle of mosquitos that spread the pandemic along the various cities [1], and look for other hidden feature to be exploited.

Another point to be aware of is about the forecasting models: one of them is **ARIMA** [6], in which lag windows are used to retrieve previous informations to let the regressor taking account of past data. Another way to work with the temporal analysis is to exploit Recurrent Neural Networks, in particular **LSTM**.

REFERENCES

[1] G. Chowell, C. Torre, C. Munayco-Escate, L. Suarez-Ognio, R. Lopez-Cruz, J. Hyman, and C. Castillo-Chavez, "Spatial and temporal dynamics of dengue fever in peru: 1994–2006," *Epidemiology & Infection*, vol. 136, no. 12, pp. 1667–1677, 2008.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, vol. 112. Springer, 2013.

[4] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), pp. 785–794, ACM, 2016.

[5] A. V. Dorogush, A. Gulin, G. Gusev, N. Kazeev, L. O. Prokhorenkova, and A. Vorobev, "Fighting biases with dynamic boosting," *CoRR*, vol. abs/1706.09516, 2017.

[6] S. C. Hillmer and G. C. Tiao, "An arima-model-based approach to seasonal adjustment," *Journal of the American Statistical Association*, vol. 77, no. 377, pp. 63–70, 1982.