Machine Learning - Supervised

Rafael Barón González - Ripoll
UCLM
Ciudad Real, Spain
Rafael.Baron@alu.uclm.es

Samuel González Linde
UCLM
Ciudad Real, Spain
Samuel.Gonzalez3@alu.uclm.es

Author Keywords
Machine Learning; Decision Trees; Regression; kNN; Dengue

INTRODUCTION

The purpose of this work is to explore the environmental data collected by various U.S. Federal Government Agencies from two cities (San Juan, Puerto Rico and Iquitos, Peru) to gain a better understanding of the Dengue Spread Phenomena. These data are from a competition of the site DrivenData.

The task is to predict the number of dengue cases for each week in each location (San Juan and Iquitos) based on those environmental variables. In order to achieve this goal we are going to use some supervised learning techniques such as Decision Trees, Linear Regression and K-Nearest Neighbors.

DATASET

The Dataset contains environmental data from the cities of San Juan and Iquitos.

The information was collected by weeks from the year 1990 to 2010 and it contains variables such as the humidity, maximum and minimum temperatures, precipitations, normalized difference vegetation index...

We are going to have 3 different datasets, two of them for training and one of them for testing and generating the prediction which is going to be uploaded to the competition page.

Training datasets will contain the values and also the total number of dengue cases for each week. On the other hand, testing dataset contains just the environmental data that we need to predict the cases for each week.

Algorithms

In order to create the prediction models, we are going to use the algorithms of Linear Regression, Decision Tree Regression and KNN, paying special attention to the first two.

Between Linear Regression and Decision Tree we decided to focus more on the Decision Tree algorithm as it is a type of algorithm with which we have not worked as much as the other could be.

For KNN we are going to apply the whole preprocessed dataset. The other two algorithms are going to be applied for two different datasets:

- Dataset as a whole with both cities
- Dataset splitted into two different parts: one with the San Juan environment data and the other with Iquitos'

In both cases the dataset will go through a previous preprocessing process.

Preprocessing Data

First of all we are going to check for null values inside the dataset. In our case we are going to fill every NaN value by propagating the last valid observation forward to next valid backfill.

Dropping features that we are not going to use is a good idea to get rid of some non usefull information. In this case we are going to drop the week_start_data as it contains time values.

As we have a field with string data, we need to prepare them by encoding them into values easier for the algorithm to be used. In this case we can use codifications such as LabelEncoder, OneHotEncoder or LabelBinarizer.

We are not going to use Label Encoder because, as we learned in Data Mining, it assign integers values for each data which implies order and that can skew the algorithm since the distance between the different coded values is greater for some values than for others.

Roughly speaking we have not found a major difference between the other two encoders as both uses 1s and 0s to encode the data but OneHotEncoder gives us a float matrix and LabelBinarizer gives us a integer matrix, so we have decided to go for the LabelBinarizer option.

Feature Selection

In the set of features that we select we think is important to have a *time* number, in both cases for the Linear Regression and the Decision Tree we choose to use the week of the year. We choose it because for a regression that evolves with time it good to attach in some way the rest of the variables to a timeline.

As we will see when we correlate all the features some are more directly correlated than other, being some not even correlated at all. We have choosen those variables based, some of them, in experimentation between all of the features provided by the dataset.

LINEAR REGRESSION

Linear Regression Algorithm is a machine learning algorithm based on supervised learning. It is a part of regression analysis which is a technique of predictive modeling that helps on finding out relationship between input variables and target values.

In this case, it represent the simplest implementation of our tested algorithms, and it gave us results very quickly. The problem is that, mostly, the only thing we can do is to select different features and run it, but it doesn't have very much left to do.

Linear Regression (Both Cities)

Our first attempt to execute the algorithm was by using the dataset as a whole, which implies that the same features have been used for the two cities.

Feature Selected

The following features were the selection that we tried for the algorithm: **weekofyear**, **reanalysis specific humidity g per kg**, **station avg temp**, **reanalysis tdtr k**.

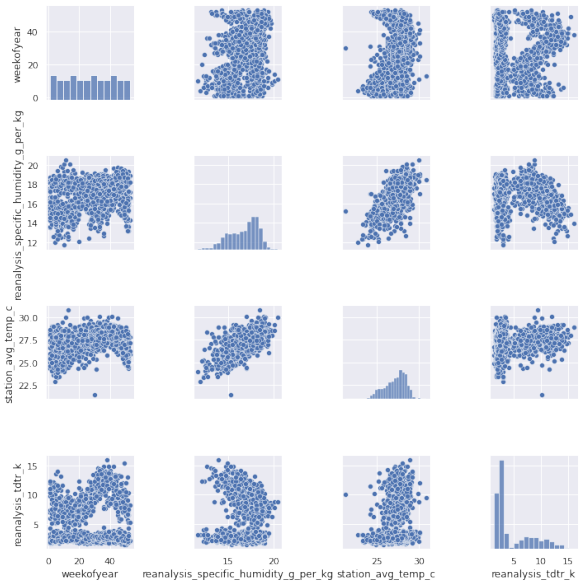

Figure 1. Features' pairplot

Algorithm

Once we run the algorithm we will obtain the prediction results from the testing dataset. We also will have access to the regresion coefficient.

The regression coefficients are estimates of the unknown population parameters and describe the relationship between a predictor variable and the response. Those values multiply the predictor values and indicates the direction of the relationship between a predictor variable and the response variable.

In the Figure 2 we can see the values assigned for each of the selected values.

	Feature	Coefficient
0	weekofyear	0.597123
1	reanalysis_specific_humidity_g_per_kg	-1.258076
2	station_avg_temp_c	4.784638
3	reanalysis_tdtr_k	-3.987188

Figure 2. Regression Coefficient

Linear Regression (Divided by Cities)

Finally our last attempt was to divide the dataset by the two cities that appears on it, and make a separate prediction between both and joining them in the final result for the test on the web page. This decision was made to take advantage on using different features for each prediction as the cities may have different circumstances in their environment data.

Features Selected

San Juan

The following features were the selection that we tried for the algorithm: **weekofyear**,**station min temp c**, **reanalysis specific humidity g per kg**.

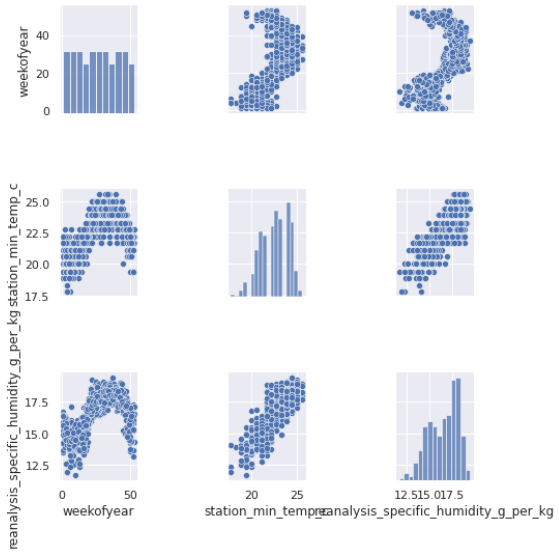

Figure 3. Features correlation for San Juan

	Feature	Coefficient
0	weekofyear	0.869083
1	station_min_temp_c	-0.828518
2	reanalysis_specific_humidity_g_per_kg	2.593672

Figure 4. Regression Coefficient for San Juan

Iquitos

The following features were the selection that we tried for the algorithm: **weekofyear**,**station min temp c**, **reanalysis specific humidity g per kg**, **ndvi sw**.

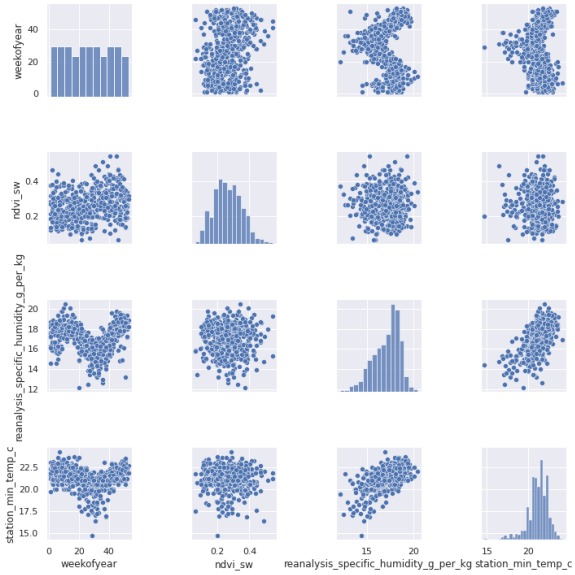

Figure 5. Features correlation for Iquitos

	Feature	Coefficient
0	weekofyear	0.000773
1	ndvi_sw	5.123620
2	reanalysis_specific_humidity_g_per_kg	1.319141
3	station_min_temp_c	0.822331

Figure 6. Regression Coefficient for Iquitos

DECISION TREE REGRESSION

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller subsets creating a tree with decision nodes which represents the values for the attribute tested and leaf nodes which represents the decision on the numerical target.

They are used to fit a sine curve with addition noisy observations. We can also set a value for the maximum tree depth which, by using higher values, can lead to overfitting. In our case, the data have noise and 'random' super-high values. So we thought this method could provide useful results.

Decision Tree (Both Cities)

Features selected

The following features were the selection that we tried for the algorithm: **weekofyear**, **reanalysis specific humidity g per kg**, **station avg temp**, **reanalysis tdtr k**.

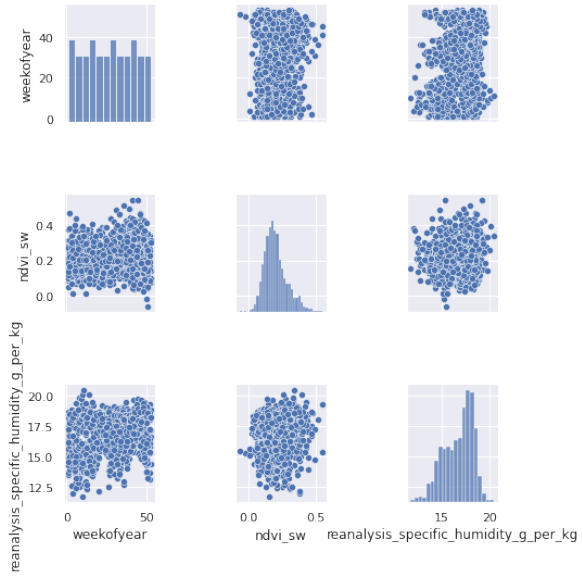

Figure 7. Features' pairplot

Decission Tree (Divided by Cities)

As well as we did in the Linear Regression we have divided the dataset by the cities so we can use different features for the predictions.

Features Selected

San Juan

The following features were the selection that we tried for the algorithm: **weekofyear**, **reanalysis specific humidity g per kg**, **ndvi sw**.

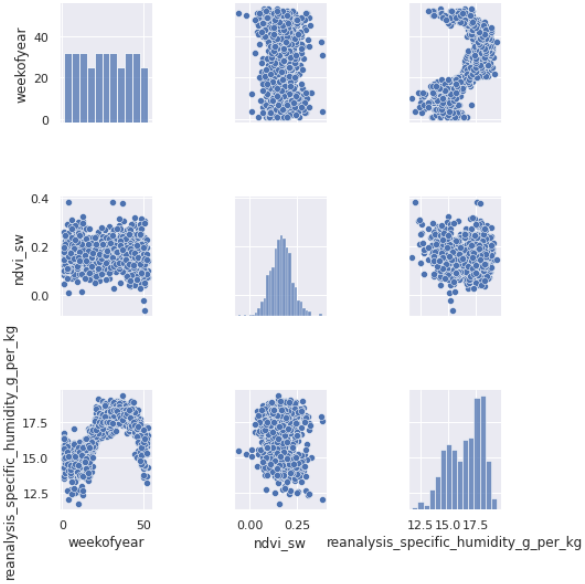

Figure 8. San Juan correlation between features

Iquitos

The following features were the selection that we tried for the algorithm: **weekofyear**, **reanalysis specific humidity g per kg**, **station min temp**, **ndvi sw**.

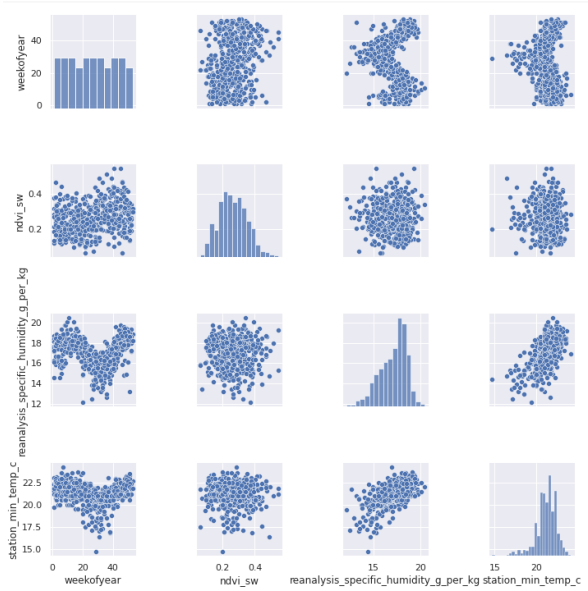

Figure 9. Iquitos features correlation between features

Feature Importance

Once we have applied the algorithm to the selected features we can obtain the importance for each feature. The feature importance refers to the assigned score for that feature based on how useful it is while predicting a target variable.

Taking a look at the Figure 10 we can see that most of the feature importance is focused in one of the selected values.

We were looking for other different combination of variables and even increasing the number of them in order to find a better distribution in the importance value but the result that it generated in the final solution was giving us a worse score than before. So finally, we decided to continue with those values.

	Feature	Decision Tree
0	weekofyear	0.214504
1	ndvi_sw	0.722324
2	reanalysis_specific_humidity_g_per_kg	0.063173

Figure 10. Feature Importance for San Juan

In the Figure 11 we can appreciate the importance value for each of the features selected from the Iquitos dataset.

The value for the importance is distributed through every feature in a stable way except for the humidity which takes a slightly higher value.

	Feature	Decision Tree
0	weekofyear	0.233420
1	ndvi_sw	0.196276
2	reanalysis_specific_humidity_g_per_kg	0.441176
3	station_min_temp_c	0.129128

Figure 11. Feature Importance for Iquitos

Subset of training data

We think is good to compare real data to out predictions, but the data providaded for testing doesn't include the data we need to make comparations. For that reason we split the training data given into a subset for training and testing.

The way to split the original train data into a new set of training and testing choosed was to use the *train_test_split()* function from *sklearn.model_selection* without any shuffling.

Mean Absolute Error testing

In order to know the best depth of the tree to generate, instead of just guessing we did run some test to see how the Mean Absolute Error (from here called MAE) evolves for different depths of the tree.

To get to do that we need actual data that the test data the competition provides does not have. Because of that, we divided the train data provided and used about 80% for training and the rest for test. With this subset of data we can calculate the MAE. In the case of the city of San Juan we get the next result:

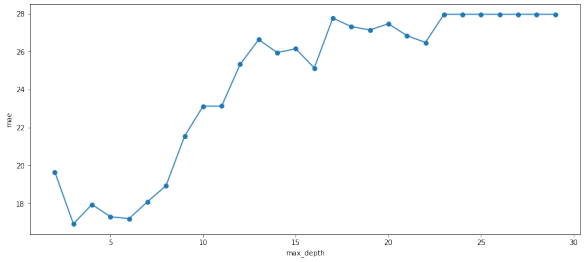

Figure 12. MAE from subset of San Juan

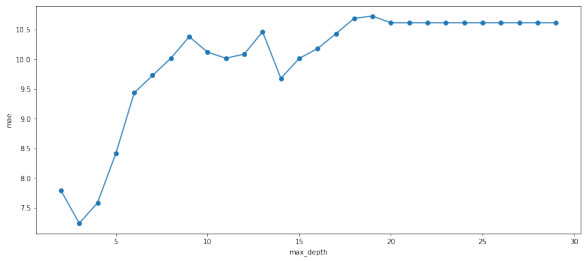

Figure 13. MAE from subset of Iquitos

As we can see in the figures, the more depth the more error we have in both cases, so shorter depths it works better. At some

point adding more depth doesn't change anything, in about 20-25 maximum depth.

We also saw that more or less the amount of features is linked to the MAE for the depths, if you go too far, it increases.

Using the subset to see how it compares

To take advantage of the splited original training data we can see how our predictions compares with the actual data from the training dataset. As we can see here, with the selected features, we get a good result for the city of San Juan.

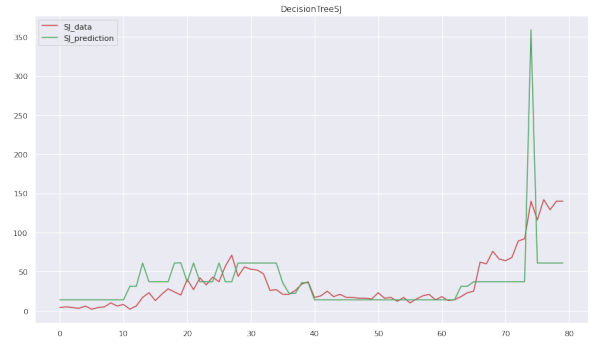

Figure 14. Prediction of San Juan using subset from the original training dataset

K-NEAREST NEIGHBOURS

We also tried a basic implementation for the kNN algorithm. To estimate the number of neighbours we also used the MAE for a given range of neighbours. With a minimun value of aproximatly 21 for the MAE with about 11 neighbours we ran the model to fit and make a prediction. This show how the MAE evolves in this case:

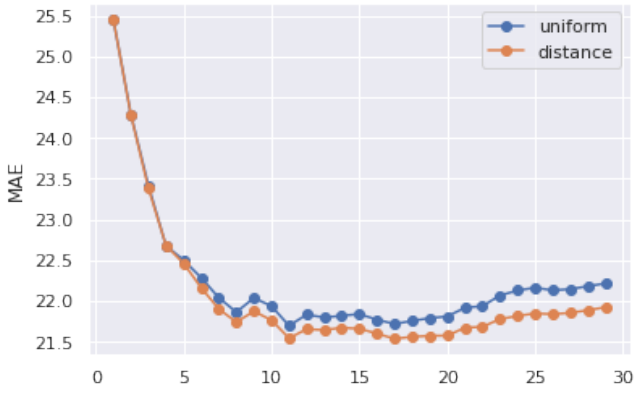

Figure 15. MAE from kNN implementation

The following features were the selection that we tried for the algorithm: **weekofyear**, **reanalysis specific humidity g per kg**, **station min temp**, **ndvi sw**.

COMPETITION SCORES

Let's see the best scores that each algorithm has left us for each case. Keep in mind that the features that have been selected are those that have been mentioned previously in each section.

Linear Regression

Both cities: 28.0649

Divided by cities: 27.0625

Decision Tree Regression

Both cities: 25.2788

Divided by cities: 25.1587

K-Nearest Neighbours

Both cities: 30.6106

CONCLUSION

Decision Tree Regression seems to be a good option for prediction with the given data set, although a broader investigation of both cities and their circumstances would be necessary in order to better understand the data provided to us. Also a better optimization of the data would help to achieve a higher score.