EURECOM
Sophia Antipolis

Challenge 1 Report
Weather Prediction

Federico Germinario, Andrea Ghiglione, Gabriele Gioetto, Nour Jamoussi

10-05-2022

Abstract

In this report we present an approach to a regression problem which consists of predicting temperature values around the globe. Our approach is based on data analysis, feature engineering and model selection in order to extract knowledge from the raw data. The proposed approach provided satisfactory results overcoming a naive baseline.

1 Problem Overview

1.1 Introduction

The project consists in the definition of a model for the prediction of the air temperature at 2 meters above the ground at different locations. The input to our model is a collection of features consisting of several meteorological features like the sun elevation, climate temperature, pressure, topography and multiple meteorological parameters besides some other features generated by Global Forecast System (GFS), Global Deterministic Forecast System (CMC) and Weather Research and Forecasting (WRF). Our aim is to obtain a model well evaluated by the negative mean squared error on the Kaggle platform.

1.2 Data Exploration

The whole dataset consists of a training set and a test set:

- a *training* set, containing 1993574 records for which it is present a temperature measured in Celsius degrees.
- a *test* set, composed by 552174 records.

In addition to the numerical target variable of the training set, expressed as the temperature, each record of the sets has 111 numerical features, which, as we said, are either generated by forecast models or taken as measurements, in particular:

- *fact time*: the unix timestamp in which the measurement has been taken.
- *geographical position*: latitude and longitude positions of the record and topography or bathymetry of the location.
- *sun elevation*: elevation of the sun.
- *climate features*: temperature and pressure.
- *model generated features*: features generated by the three different weather prediction models (GFS, CMC, WRF). The features consist of values of wind, pressure, humidity, temperature, precipitations at different height and pressure level.

Considering the training and test set, it is noticeable that the locations from which the measurements are taken partly differ, in particular in the test set there are some points coming from South America, Africa, India and Oceania which are not present in the training set.

Having to deal with a lot of similar features coming from weather predictions models and measured on different pressure and height level, we can also expect a good percentage of features to be correlated as well as missing values and incompleteness of the data.

Finally we can have a look at how the temperature is changing across the world in Figure 1 and we can visualize its distribution in Figure 2.

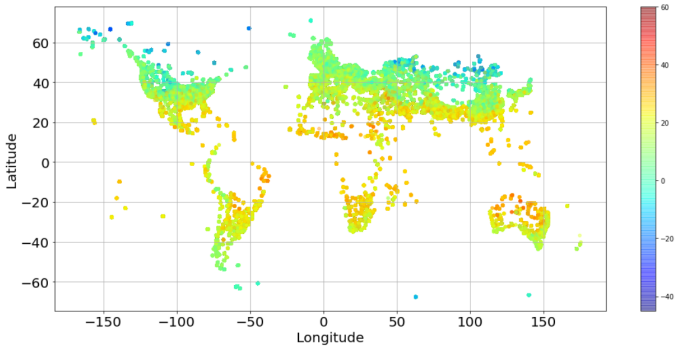

Figure 1: Temperature distribution across the globe.

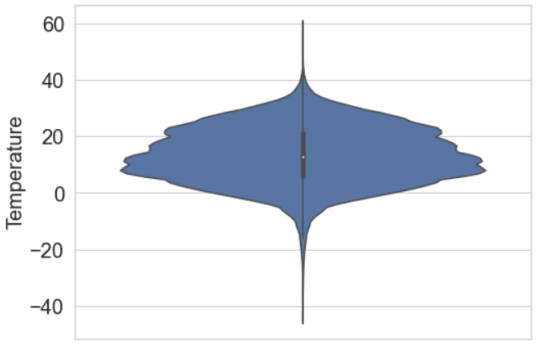

Figure 2: Violin plot of the temperature.

2 Proposed Approach

2.1 Data cleaning

First of all, we had to deal with the null values which are present in the training set. In particular this set contains 108868 records which have some feature with a *NaN* value. We noticed that all the null values are corresponding to features related to two weather forecast models; CMC and WRF. As we can see from Figure 3, a big amount of missing data is from West of Australia, hence we can not simply delete the records with missing values, since we would lose a relevant amount of information (roughly 5% of the whole training set). In order to fill the missing values we proceeded as follow: we first sorted the training set in terms of latitude and longitude, and then we applied a backward and forward filling. In this way we were able to infer data based on geographical position on the globe. Once we did that, we have dealt with the problem of *duplicated data* in the training set. There are some cases in which we should keep the duplicates in the dataset, because the same information appearing twice might be itself an information. However, in this case there is no reason for which we should keep duplicates; the same weather information at the same timestamp is a clear example of misleading information. In addition to that, by removing duplicates our model will be able to better generalize the dataset. For the reasons that we mentioned, we have decided to remove the duplicates.

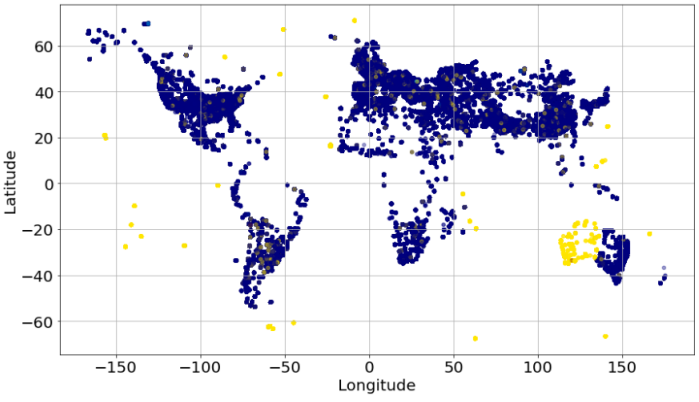

Figure 3: Overview of missing data; yellow, gray and red points corresponds to respectively WRF, CMC and both missing values. Blue points are the not null points.

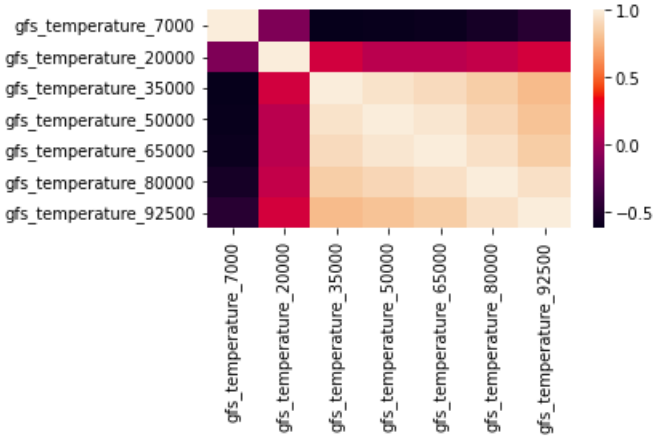

Figure 4: Correlation Matrix of some GFS temperature features

2.2 Feature selection

Since we had to deal with many features we decided to perform feature selection in order to lighten the training and to only exploit useful features. We considered visualizing correlation inside groups of features and we noticed that there was high

correlation among them. In particular this happened in set of features coming from weather forecast models (GFS and CMC). An example of these correlations is reported in Figure 4. For this reason we removed features which are highly correlated with other features. During our experiments we also tried feature selection using forward selection and Principal Component Analysis which however did not provide better results.

2.3 Feature engineering

As a further step we performed feature engineering on the *fact time* attribute which is the unix timestamp of the temperature measurement. In particular we extracted the information about the month and the hour of the day in which the measurement was done; the hours were split into 6 bins. The two information are more useful than the timestamp because they have a more direct impact on the temperature since months characterize seasons and the hour characterizes the moment of the day. However, we did not perform a simple retrieval of this information because otherwise we would have introduced a misleading relation between these information and the target variable (for example, a bigger month doesn't imply an higher temperature). For this reason we applied a *one hot encoding* in order to add new information and no misleading relations among features in the dataset. In addition, this method did not add a lot of features so the sparsity problem was not relevant and we did not fall in the *curse of dimensionality* problem. We also tried *outliers removal*, however visualizing the temperature data on the map it was clear that there were no absurd measurement and so removing outliers would have meant removing useful information about very cold and very hot spots in the world (which are also useful since there is a distribution shift in the test set, with respect to the training set). For this reason we did not perform outliers removal. Analyzing the dataset we have seen that all the features are numerical. Thus, there was no need to perform additional data encoding.

Next, to prevent the optimization from getting stuck in local optima and to make the training faster, we performed a *feature scaling*. For that, we used the Standard Scaler implemented in scikit-learn library.

2.4 Model Selection and hyperparameters tuning

We have tried two different regression models:

- Linear regression with Lasso: between the two, it is the simplest model. We have chosen this model since we still have a high number of features and with L1 regularization the model sets some features weight to zero, thus eliminating them. Therefore we do an additional feature selection implicitly. We performed hyperparameters tuning working on the regularization strength following a coarse-to-fine grid search approach as reported in Table 1.

α **- coarse**	1	0.1	0.01	0.001	0.0001
α **- fine**	0.001	0.003	0.005	0.007	0.009

Table 1: Hyperparameters tuning for Linear regression with Lasso

- Multilayer perceptron: the MLP is able to learn non-linear functions, however a large amount of data is needed in order to get good results. Luckily, we have about two million training data, which should be enough. The parameters were initialized using the *He-Normal Initialization* to speed up the convergence, and after each level of the MLP we added a batch normalization layer to deal with the internal covariance shift problem and with the vanishing gradient problem. The activation function used after each layer was the *ReLU*. We did not add dropout, since the MLP already generalized well on unseen data.

3 Results

3.1 Numerical results

Linear regression with Lasso turned out to work better with a regularization strength value in the order of magnitude of 10^{-3} from the coarse grid search. After the fine hyper parameter tuning, the best α parameter was again 0.001. The performance with this model setting allowed us to achieve a result of 0.258 in terms of mean squared error on the validation set.

Multilayer perceptron lets us achieve a better result with respect to Lasso. In particular we obtained a mean squared error on the validation set of 0.0408. During the training the validation loss followed a decreasing path, as shown in Figure 5, which is a good sign in term of limiting the overfitting. Both training and validation losses didn't decrease much after the epoch 25, even though we set *early stopping* criteria to monitor if the validation loss did not decrease for more than 3 epochs. This happens because the validation loss decreases very slowly, even though in the plot it seems constant.

3.2 Limits and merits of the models

Linear Regression was mainly used as a baseline since it can only model linear relations. We could have overcame this problem by computing polynomial features. However, the high number of features discouraged us to use this approach. We have to highlight that this model uses few parameters, making the training fast.

One of the main drawbacks of MLP is that we need to set a number of layers and units per layer, which may be hard to fine tune since this kind of model can be computationally expensive and time consuming. On the other hand, MLP can model complex functions providing satisfactory results.

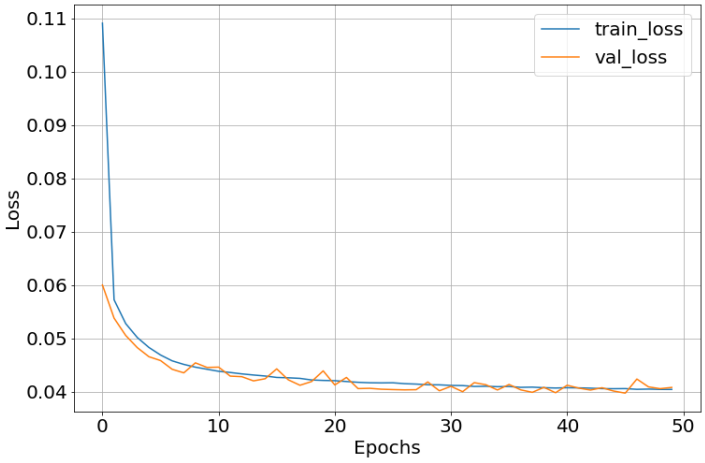

Figure 5: Training loss and validation loss - MLP