Technical Report Challenge 1 - Weather Prediction

Algorithmic Machine Learning Course (AML), Spring 2023, EURECOM

I. Introduction

The Weather Prediction problem at hand consists of dealing with pairs of meteorological features and target values at a particular latitude/longitude and time. The task is affected by the mismatch between training and deployment data. This is due to training data sampled from a distribution, which is different from the one where test data is drawn from. More specifically, while the distribution of inputs may change over time, the labeling function, i.e. the conditional distribution $P(y|x)$, does not change. This phenomenon is known as covariate shift because it arises due to a shift in the distribution of the covariates, namely the features.

II. Data analysis

A. Data exploration

The dataset consists of 113 heterogeneous features, i.e. of different types and scales that are highly correlated among each other, and it has a total of 1993574 records. The feature vectors include weather-related features, which are either the result of direct measurements, or of weather forecast prediction models, namely Global Forecast System (GFS), Global Deterministic Forecast System from the Canadian Meteorological Center (CMC), and the Weather Research and Forecasting (WRF) Model. Each of the mentioned models returns the following predicted values: wind, humidity, pressure, clouds, precipitation, dew point, snow depth, air, and soil temperature characteristics. These are given at different isobaric levels from 50 hPa ($\approx 20km$ above ground) to the ground level. The target variable is the air temperature measurements at 2 meters above the ground. At first glance the Gaussian-like shape of the target variable distribution stands out. This histogram is shown in Figure 1. This feature allows us to accurately describe the distribution of heterogeneous values thanks to assumptions that can be specified by its mean and standard deviation. If those two are known, every data point is bound to be easy to access on the curve, making the math straightforward.

The correlation among all pairs of features was studied by evaluating the correlation matrix and plotting its heatmap, which is shown in Fig. 2. Some features are highly correlated to other features belonging to the same weather forecast model, e.g. all the "gfs_temperature_xxxxx" features. Moreover, certain features are highly correlated to features outside of their weather forecast model, as, for example, for the "cmc_0_3_5_xxx" columns, that represent the geopotential height at different pressures, which are positively correlated to the "gfs_temperature_xxxxx" features. Finally, several features ("climate_temperature", the "gfs_temperature_xxxxx" features, "wrf_t2_next" and "wrf_t2_interpolated" etc..) are highly correlated to the target variable.

Most of the latter are estimates of the temperature according to the different weather forecast models, thus it makes sense that they are highly correlated to the target temperature; furthermore, provided that the models are proven to be trustworthy, these features can be considered reliable predictors for our task.

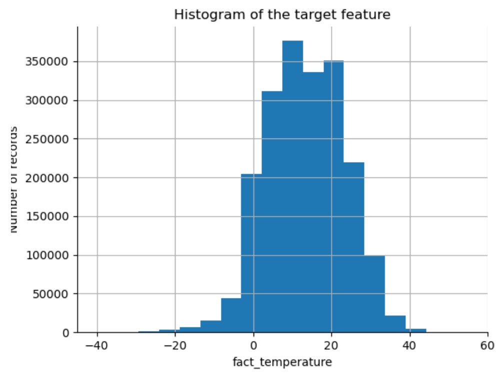

Fig. 1. Histogram of the target feature, fact_temperature

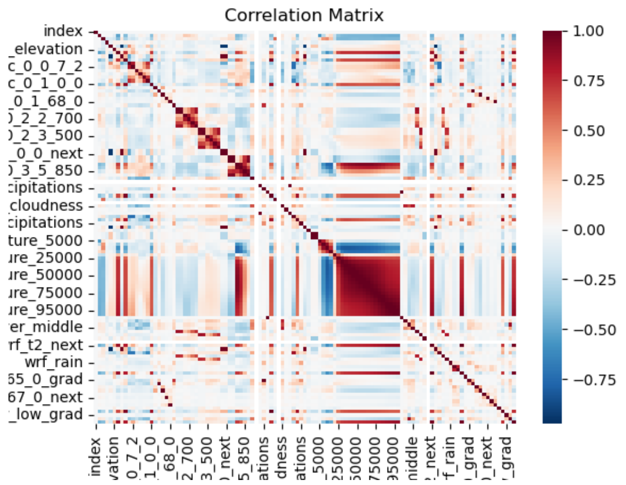

Fig. 2. Heatmap of the correlation matrix of the training dataframe

B. Missing values

The dataset presents missing data: the histogram in Fig. 3 shows the distribution of missing data per column as a percentage. As can be seen, most features have zero or almost zero missing rows, with only 10 columns having $\approx 5\%$ of their rows missing.

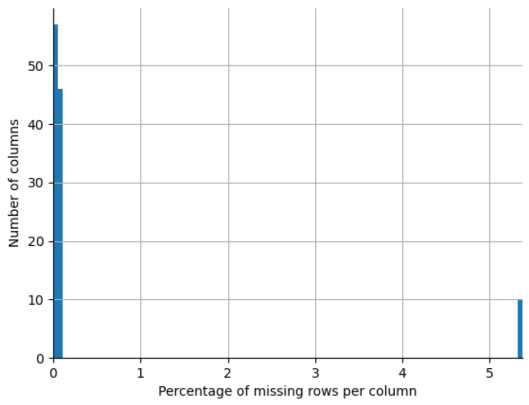

Fig. 3. Histogram of the percentage of missing data per column

The matrix visualization of the missing data in the dataframe, shown in Fig. 4, is a useful tool to decide how to handle it. As can be seen in the matrix plot, there is a group of adjacent columns with missing data in some of the rows, corresponding to those with $\approx 5\%$ of missing rows shown in the previous histogram, and then there is a small group of rows with missing data in several columns. Our approach in dealing with NaNs is outlined in the dedicated subsection under "Data Pre-processing".

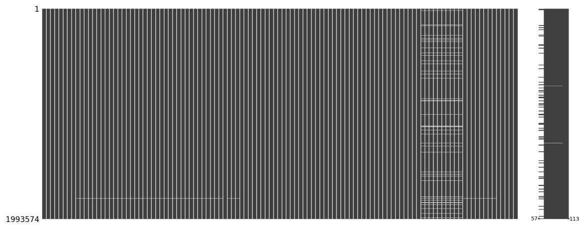

Fig. 4. Matrix visualization of missing data in the training dataframe. When data is present, the plot is shaded in grey and when it is absent the plot is displayed in white.

C. Outliers

A further concerning factor that needed to be addressed was the existence of points that differ significantly from other observations present in the given dataset, known as multi-variate outliers. Therefore, to estimate their amount with respect to the total number of available samples, two techniques have been adopted: Inter Quartile Range (IQR) and Z-score. The former counts a value as an outlier if it is placed outside the range between $-1.5xIQR$ and $1.5xIQR$, whilst the latter assesses the number of standard deviations an observation is away from the mean of the variable distribution, which is assumed to be Gaussian. The Z-score has been employed to quantize the number of outliers per feature, by fixing a threshold of three standard deviations from the mean, above which the samples are considered as outliers. This led to the finding that each feature contained a significantly high number of outliers. Alongside with this consideration, the nature of weather data itself, which can contain important information about extreme weather events, contributed to deciding not to discard any outlier for the sake of the substantial information they carry, despite their deviation from the average.

III. Data pre-processing

A. Missing values approach

The small group of rows with many missing columns was dropped.

Upon further investigation, it was found that the group of columns with $\approx 5\%$ of missing data corresponds to all the features belonging to the WFS weather forecast model. They are either all present or not at all for a given row, and the presence of their values or lack thereof is also specified by the "wfs_available" column. Given that 3 out of these 10 features are highly correlated with the target variable, and given that the percentage of missing rows for these columns is small, we decided to drop the rows with missing data on them. This decision was taken also because interpolation was ruled out as a method to deal with the missing values since interpolating the values associated with a given forecast model was deemed unreasonable. Also, filling them with a constant value, such as 0 or the mean of their respective column, was excluded, as it would have modified the distributions of values in such columns.

B. Train-validation split

The training dataframe is split into training and validation data and labels, with 30% of the rows being chosen with shuffling for validation.

C. Normalization

The data is standardized, i.e. each value is subtracted by the mean value of its column and divided by its standard deviation, so that each column ends up having zero mean and unit variance. The *StandardScaler* class from *scikit-learn* is employed: a StandardScaler object is fitted on the training data and it is then used to transform both the training and validation data. The same object is also used later to standardize the test data, so that no bias is introduced. Moreover, the target variable of both training and validation data are standardized using the mean and the standard deviation of the training variable. It follows that, at inference time, the predictions must be un-standardized using the same statistics.

D. *Feature selection*

We observed that three features had zero variance, meaning that they were constant: "cmc_available", "gfs_available" and "wrf_available". These categorical variables indicate whether the columns associated to their respective weather forecast model contain data for a particular row or not. Since they are constant and they give no further information about the samples, these columns were dropped.

Furthermore, removed those variables that are redundant in order to decrease the dimensionality of the dataset and help speed up processing. In our analysis, a variable is deemed redundant if it has a high correlation (either positive or negative) with another variable, which is a sign that two such features express the same information. We chose a rather high threshold of 0.99, meaning that the two features correlate very closely. 29 features that correlate with some other variable were found above this threshold, and for each group of correlated features, we removed features so that only one remained per group. 13 columns in total were removed using this technique.

Finally, univariate feature selection was performed evaluating the Pearson's correlation coefficient of each feature, which is a measure of correlation between a numerical variable and a numerical target. Given that $\approx$ 20 of the remaining features scored considerably higher than the others on this metric, we decided to keep the 30 features with the highest values of Pearson's coefficient, which greatly reduced the dimensionality of the dataset while still keeping the most "informative" features.

IV. Data shift correction

The lack of training samples in the equatorial area indicates a noticeable shift in the distribution of samples between the training and test sets. This translates into observations x_i drawn from some source distribution $q(x)$, rather than the target distribution $p(x)$. This phenomenon has also been demonstrated by iteratively applying the two-sample Kolmogorov-Smirnov test for goodness of fit along the columns. The test successfully compares the underlying continuous distributions of two independent samples [1] .

Under dependency assumption, the conditional distribution does not change, which means that $p(y|x) = q(y|x)$. Hence, source distribution $q(x)$ could be corrected via a weighting technique. More specifically, this approach takes place by exploiting this identity in the expected loss, inferred via the conditional probability:

$$\iint l(f(x), y)p(y|x)p(x)dxdy = \iint l(f(x), y)q(y|x)q(x)\frac{p(x)}{q(x)}dxdy$$

For further insights on the mathematical intuition, please refer to the Source [2]. Therefore, to tackle this issue, each data example has been re-weighted by the *density ratio*, i.e. the ratio of the probability that it would have been drawn from the test distribution to that from the train one:

$$\beta_i = \frac{p(x_i)}{q(x_i)}$$

This correction has been achieved by concatenating train and test data in order to train a Logistic Regression binary classifier. The classifier in question has been trained on standardized data, whose top-15 most relevant features have been retained via Analysis of Variance (ANOVA). Subsequently, the density ratio for the train set has been computed. Lastly, plugging in the obtained weight for each data example (x_i, y_i), the selected model has been trained using weighted empirical risk minimization:

$$minimize_f \quad \frac{1}{n}\sum_{i=1}^{n} \beta_i l(f(x_i), y_i)$$

This led to re-scaled training points sizes by their weight. As a result, the training points shown in Figure 5 appear not to be of the same dimension among each other as in the original configuration. The ultimate goal of this covariate shift correction is to put more weight on the samples with feature values that are more frequently shown in the test set, and vice versa.

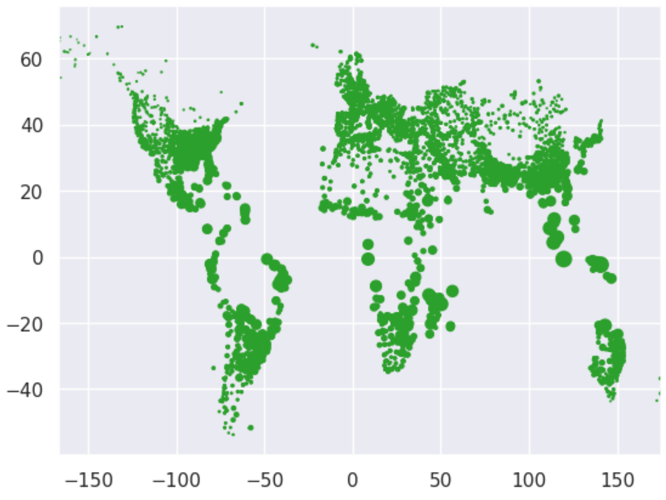

Fig. 5. Coordinates of scaled train points in the dataset

V. Modeling selection

A. *Baseline model*

To have an initial guiding metric, we used a simple Lasso regressor as a baseline model so we could compare our results.

B. *Other models*

Gradient boosting-based models were trained and evaluated, which are based on an iterative procedure (*"bosting"*) that builds an ensemble model by adding decision trees serially so that each new one tries to learn the "residual", i.e. the error of the previous in order to minimize the objective function at every step, just as in the gradient descent algorithm. The final prediction is a weighted sum of all of the tree predictions.

XGBoost is a scalable and highly accurate implementation of gradient boosting, which aims at achieving the best performance while optimizing for computational speed on machine learning problems (See [3]). It has consistently been shown to be one of the best machine learning algorithms available, and we obtained the best results on this challenge with it.

CatBoost was also employed for its ability to automatically handle feature scaling, which can be beneficial when dealing with features on different scales, and for its computational complexity and time benefits (See [4]).

Finally, we also trained a Random Forest regressor, because of the ability of random forests to handle well heterogeneous data and because of the possibility to parallelize their training.

VI. Results

A. Baseline model

A Lasso regressor was trained with hyperparameter $\alpha = 1$. It achieved the following results:

Training RMSE :	9.985
Validation RMSE :	9.999

The regularization coefficient used is so high that the final predictions are constant, meaning that the learned function is of zeroth order.

B. Random Forest

Training time increased substantially with an increasing number of estimators used. A few combinations of the *"max_depth"* and *"n_estimators"* parameters were tried, and the best result obtained is:

max_depth :	10
n_estimators :	250
Validation RMSE :	2.341

Not many attempts were made with this model since much better results were achieved out-of-the-box with XGBoost, which also proved to be more computationally efficient.

C. XGBoost

Several XGBoost models were trained with different combinations of the *"max_depth"*, *"n_estimators"* and *"eta"* parameters. The best result obtained is:

max_depth :	12
n_estimators :	2000
eta :	0.1
random_state :	42
Validation RMSE :	1.798

D. CatBoost

The *grid_search* function of the *CatBoostRegressor* class, from the *catboost* library, was used to test the following hyperparameters:

iterations :	100,	150,	200,	300
learning_rate :	0.03,	0.1		
depth :	4,	6,	8,	10
l2_leaf_reg :	0.2,	0.5,	1,	3

The best result obtained is:

iterations :	300
learning_rate :	0.1
depth :	10
l2_leaf_reg :	0.2
Validation RMSE :	2.204

Given that the training process for the CatBoost model was noticeably faster than for XGBoost, an attempt of training with *n_iterations*=10000 was made. The obtained RMSE score on the validation set is 1.838, which did not match the best performance obtained with XGBoost.

VII. Conclusions

In conclusion, the best results were obtained by the XGBoost model. Since we expected to see it outperforming the other models, we focused our efforts on feature selection. This helped to improve the computational complexity but probably did not contribute that much to the final performance, since there were many features that correlated highly with the target variable. We also focused on handling the covariate shift between training and test data, to prove that a straightforward approach to tackle it can be easily implemented within a traditional pipeline for a machine learning problem.

References

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ks_2samp.html

[2] https://d2l.ai/chapter_linear-classification/environment-and-distribution-shift.html

[3] https://www.nvidia.com/en-us/glossary/data-science/xgboost/

[4] https://catboost.ai/