Rainfall Prediction

Alae KICHI, Jinan LOUBANI

January 2022

Contents

1 Presentation of the problem

In this part, we give a general presentation of the problem being investigated. Section 1.1 presents the context of the problem. Section 1.2 states the main objective. Section 1.3 specifies the data sources.

1.1 Context

This work aims at predicting the accumulated daily rainfall on the D day on the observation ground stations for Meteo France. The data we investigate is MeteoNet which is an open reference weather data set for Artificial Intelligence. The zone we consider is the North-West quarter of France and the period is 2016-2019.

1.2 Objective

The main purpose of this work is to develop an AI model that outperform the existing models of Meteo France based on physical laws and data assimilation techniques.

1.3 Data

The data sources we have are of two types: measurement stations and weather forecasts. For measurement stations, we have different features (temperature, wind...) measured at a timestamp t. The weather forecasts are made by weather forecast systems from Meteo France and they predict the weather characteristics in the future. The idea is to use the available data (measurements of the day $D-1$ and weather forecast of the day D) to predict the rainfall.

2 Tools and infrastructure used

In this part, we present the list of tools we used in solving the problem. The tools and infrastructure we used in preparing data and training models are the following:

- Python: we used python as the main programming language in preparing and training data.
- Plotly: our main visualization tool was plotly.
- Networkx: we used the python library Networkx to study the graph we used to implement missing values.
- Scikit-Learn: we used Scikit-Learn for data preparation and to train our data using its built-in function of the random forest model.
- Google Cloud: we trained and processed our models in google cloud.

- GitLab: we used gitLab based on git as a virsion control to track our work.

3 Methodology

This part is dedicated to explain the steps and methodology we followed to resolve the problem. Section 3.1 gives a general overview of the data to understand its distributions and the existing relationships among its features. Section 3.2 shows the different methods we used for data pre-processing. The three main aspects we considered are: imputing missing values, aggregating data and scaling features. Section 3.3 is about feature engineering: generating new features from the existing features. The two main axes we considered are: feature extraction and polynomial feature engineering. Section 3.4 is dedicated to the models and algorithms we tested to solve our problem. The two models we used are the Random Forest and the XGBoost models.

3.1 Data Exploration

Data exploration is an essential step before pre-processing data to be able to make the right decisions for further steps. The techniques and strategies used in data pre-processing are various and strongly depend on the distributions of the features and the types of the relationships existing among them.

The data we have concerns six features: temperature, precipitation, humidity, dew point's temperature, wind speed and wind direction. Plotting the histograms of these features (Appendix A.1), we find out that only the temperature and the dew point's temperature follow normal distributions. The box plots of these features (Appendix A.2) allow us to identify the presence of extreme values in almost all the data features. These conclusions in the data distributions led us to choose a consistent scaling and imputation methods as explained in Section 3.2.3 and Section 3.2.1.

To study the linear relationships between the features, we plot the correlation matrix and the pair plot of all the features (Appendix A.3). The score used in calculating the correlation matrix detects the linear relationships among the features. However, there are many non-linear relationships that the score does not detect. For that, we use the PPScore among to measure (Appendix A.4) the non-linear relationships between those features.

3.2 Data pre-processing

The three axes we consider for data pre-processing are: imputing missing values, data aggregation and feature scaling.

3.2.1 Missing Values Imputation

The percentage of missing values in our data varies considerably from one feature to another. The lowest percentage is five percent which corresponds to the temperature and the highest one is forty percent which corresponds to wind speed and wind direction as shown in Figure 1. The presence of huge percentage of missing values in our data may have grave consequences on the prediction quality. For that, we gave huge attention to this essential problem. To understand the problem more, we plotted the corresponding histogram, nullity matrix, endro-gram and heat-map (Appendix B.1.1) of the missing values. The histogram shows the distribution of the missing values among the features. The nullity matrix shows their temporal distribution. The endro-gram shows the relationship between the features regarding the presence of missing values.
To solve the problem of having many missing values, classical methods like taking the mean or the median may lead to loosing basic information and thus affect the quality of the model. For that, we used two different approaches: the first one using random forest and the second one using a graph.

	Missing Values	% of Total Values
wind_direction	966169	41.9
wind_speed	965744	41.9
dew_point_temp	857676	37.2
humidity	857227	37.2
precip	147787	6.4
temp	119684	5.2

Figure 1: The percentage of the missing values in the features

3.2.1.1 First approach: using random forest

MissForest is one of the most effective methods for imputing missing values. It starts by imputing all missing data using the mean/mode, then for each variable with missing values, MissForest fits a random forest on the observed part to predict the missing part. In our approach, We start from columns with the largest number of missing values to columns with fewest number. We repeat this process of training and predicting several times to impute all the missing values. For the estimation measure, we use the out-of-bag method to estimate the generalization accuracy.

Despite the fact that this methods proves to be very efficient theoretically, in our particular case, it fails at preserving the data distribution for features not following normal distribution. In view of that, we investigated another approach based on graph modelling.

3.2.1.2 Second approach: using graph model

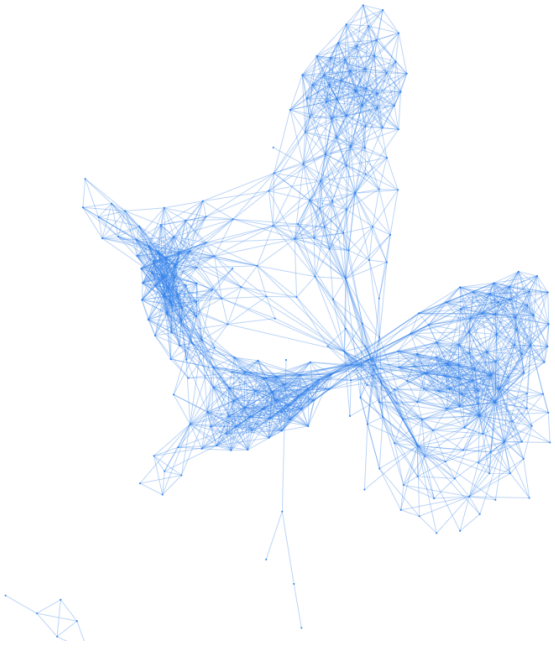


Figure 2: The graph modelling the stations

Our method in imputing missing values is to model the stations using a graph. Considering all the stations we have (even those which are not in the train set), the vertices of our graph are the stations and two stations are linked by an arc if the distance between them is less than fifty kilometers. With this model, we fix a feature and a date, and for every missing value we consider the five closest available values. Instead of taking the average of these values, we take the weighted average of them with the weights being the inverse distances using the formula [3]

$$z(x_j) = \frac{\sum_{i=1}^{n} z(x_i) d_{ij}^{-r}}{\sum_{i=1}^{n} d_{ij}^{-r}}, \tag{1}$$

where $z(x_j)$ is the expected predicted value of the station according to the weighted average of the observed stations $z(x_1), z(x_2), ..., z(x_n)$ and d_{ij}^{-r} is the weighting factor, defined as the Vincenty distance between the observation $z(x_i)$ and the value to be estimated $z(x_j)$ and r a positive real number (usually $r = 2$). In our case, we consider the closest sixteen stations and among them we select the closest five available values i.e. we take $n = 5$. Figure 2 shows the resulting graph modelling the stations.

This method is very efficient and it proves to preserve data distribution. Figure 3 shows that this method does preserve the distribution of the humidity feature unlike MissForest which completely changes it. The results for the other features are in Appendix B.1.2.

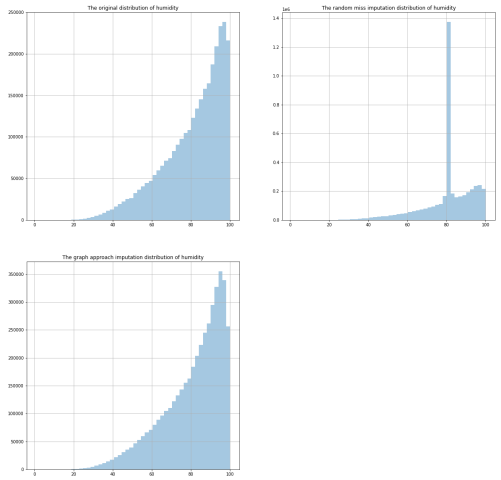


Figure 3: The percentage of the missing values in the features

Despite its perfect performance, the graph modelling method is computationally expensive due to the huge number of missing values as shown in Figure 1. To calculate the missing values of the temperature feature which has the least percentage, the computation takes 47 hours, which is not practical.

To solve the problem, we tried parallel computing using different libraries mainly multiprocessing, joblib and dask, but they were not efficient enough. For that, we converted the dataframes into numpy array and we used the library numba to parallelize our code.

3.2.2 Data Aggregation

A particular specificity of our data is that the features data are given by hours but the prediction values are given by days. To overcome this difficulty and in order not to loose information, instead or replacing the 24 values by one value (e.g. its average or median), we consider for every feature the following quantities: Minimum, Maximum, Average, Median, 1st Quartile, and 3rd Quartile. Since these quantities give full statistical description of the features, we replace every feature by these six characteristics.

3.2.3 Feature scaling

The fact that the features we have are of different nature justifies the difference in the scale of the values taken by these features. The most important advantage of feature scaling is that it prevents one significant number from impacting the model because of their large magnitude. Another important advantage is that it allows the model to converge much faster than without scaling.

The two basic types of feature scaling are:

- Normalization: bounds our values between two numbers: [0,1] or [-1,1]
- Standardization: transforms the data to have zero mean and a variance of 1.

To choose between these two types it is important to remark that mean centering does not affect the covariance matrix, but scaling and standardizing variables do affect it.

Based on these two types, the scaling techniques used are various but the mostly used ones are

- Mean Centering
- Min Max Scaler
- Standard Scaler
- Robust Scaler

Among these scaler the Robust Scaler is very interesting as it is robust to exteme values but it affects the covariance matrix. In order to preserve the initial data distribution and the covariance matrix, we chose the first scaler which is the mean centering.

3.3 Feature Engineering

The two axes we consider for feature engineering are based on: feature extraction and polynomial feature engineering.

3.3.1 Feature extraction

In our data, we have the same issue as in [5], which is that despite the fact that the wind directions 1 degree and 359 degrees are very close, numerically the difference is considered to be 358 degrees. For that, we converted the wind direction with the wind speed into x-component and y-component as expressed in [4]

$$V_x = V \cos\left(\theta \frac{\pi}{180}\right) \tag{2}$$

$$V_y = V \sin\left(\theta \frac{\pi}{180}\right), \tag{3}$$

where V is wind speed and θ is wind direction.

In addition to that, we extracted the month and season from the date feature because the weather depends obviously on the season.

3.3.2 Polynomial feature engineering

Polynomial feature engineering consists in generating new features by considering a polynomial of degree n whose variables are the old features. The generated features are the values raised to a power for each degree and the interactions between all pairs of features. After generating these features, we used the Linear Lasso Model to calculate the importance of each feature with respect to the target. By setting the threshold to be the averages of the obtained importance values, we remove the features whose importance is below the threshold.

3.4 Models and Algorithms

The existing models used for regression are various from linear model like linear regression to tree based models, discriminate models, bayses models. In our case, we chose to work with the tree based models, that are the state of art in tabular structured data: Random Forest and XGBoost. Random Forest and XGBoost models belong to the category of ensemble learning which is a technique that combines several predictors built using multiple machine learning algorithms to make a predictor having more accurate predictions than any individual predictor.

Ensemble learning is divided into two main categories: boosting and bootstrap aggregation (bagging). Boosting refers to a group of algorithms that use weighted averages of weak learners and turn it into stronger learner. It is based on the idea that one single model learns from another one which in turn boosts the learning. Bootstrap refers to random sampling with replacement. It aggregates many models together to make the final decision. The construction of every model involves random sampling of small subset of data from the dataset. The main purpose of this method is to reduce the variance for those models that have high variance, typically decision trees. In bagging, we run each model independently and then we aggregate the outputs at the end without preference to any model.

3.4.1 Random Forest

The main problem with decision trees is that they are very susceptible to random idiosyncrasies in the training dataset. We say that Decision Trees have high variance since if we randomly change the training dataset, we may end up with a very different looking tree and so different predictions as well. However, of course, decision trees do have advantages which is the reason behind trying to improve them rather than replacing them. The essential advantage of decision trees is that they make no assumptions about how the data is structured. A decision tree has the potential to get at the essence of the data no matter how it is structured. In view of that, the goal from introducing random forests is to take the advantages of decision trees while mitigating the variance issues. We mostly refer to [1] for the theoretical part of the random forest model. The random forest model was introduced to improve decision trees.

3.4.1.1 Principles

Random forest is a supervised ensemble learning algorithm. It is a bagging technique. There is no interaction between the trees constituting the forest while building them. Each tree is constructed using a bootstrapped sample which is a random sample of datapoints where we randomly select with replacement datapoints from our original dataset. To bag decision trees, we create multiple bootstrapped resamples of our training dataset.

In regression, random forest operates by constructing a multitude of decision trees at training time and outputting the mean prediction of the individual trees. A drawback of this method is that the trees could be highly correlated. To prevent this phenomenon, the following modifications are performed.

- The number of features that can be split on at each node is limited to some percentage of the total (which is known as the hyperparameter). This ensures that the ensemble model does not rely too heavily on any individual feature, and makes fair use of all potentially predictive features.
- Each tree draws a random sample from the original data set when generating its splits, adding a further element of randomness that prevents over-fitting.

3.4.1.2 Advantages and Disadvantages

The advantages of random forest are various. The main ones are the following:

- The accuracy of random forests proved to be very high on different data sets.
- It runs efficiently on large databases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates about the importance of variables in the data.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.

As any other machine learning model, random forest has some disadvantages, some of which are the following:

- Random forests suffer from over-fitting for some datasets with noisy regression tasks.
- For data including categorical variables with different number of levels, random forests are biased in favor of those attributes with more levels. Therefore, the variable importance scores from random forest are not reliable for this type of data.

3.4.1.3 Implementation

To train our data using the random forest model, we used the built-in function in the Scikit-Learn library of python. We used out of bag technique and the the squared error as a metric that the loss function minimize . For the performance evaluation, we used mean absolute percentage error.

3.4.2 XGBoost

The boosting algorithm creates new weak learners (models) and sequentially combines their predictions to improve the overall performance of the model. For any incorrect prediction, larger weights are assigned to miss-predicted samples and lower ones to samples that are correctly predicted. Weak learner models that perform better have higher weights in the final ensemble model. Boosting never changes the previous predictor and only corrects the next predictor by learning from mistakes. Gradient boosting is a special case of boosting algorithm where errors are minimized by a gradient descent algorithm and produce a model in the form of weak prediction models e.g. decision trees. EXtreme Gradient Boosting (XGBoost) is a scalable and improved version of the gradient boosting algorithm designed for efficacy, computational speed and model performance. We mostly refer to [2] for the theoretical part of the XGBoost model.

3.4.2.1 Principles

XGBoost is a gradient boosting method in which some modifications are considered to make it robust. It is based on additional penalization terms to control overfitting. XGBoost allows to optimize a lot of parameters, leading to a total control of the gradient Boosting algorithm.

XGBoost uses a particular method in pruning a regression tree to prevent overfitting of the training data. Pruning is a technique used to reduce the size of regression trees by replacing nodes that don't contribute to improving regression on leaves. XGBoost creates nodes up to maximum depth specified and starts pruning from backward until the loss is below a threshold.

XGBoost parallelizes the sequential process of generating trees. Tree learning needs data in a sorted manner. To cut down the sorting costs, data is divided into compressed blocks (each column with corresponding feature value). XGBoost sorts each block parallelly using all available cores/threads of CPU. This optimization is valuable since a large number of nodes gets created frequently in a tree.

3.4.2.2 Advantages and Disadvantages

Most of the advantages of XGBoost model can be deducted from its characteristics. The most important ones are the following:

- XGBoost has efficient execution speed and model performance in regression predictive modeling.
- XGBoost allows to optimize a lot of parameters, leading to a total control of the gradient Boosting algorithm.
- The strength of XGBoost lies in the implementation tips for parallelization, making it much faster and more accurate than traditional Gradient Boosting algorithms.
- It is often the winning algorithm of Kaggle competitions.

As any other machine learning model, XGBoost has some disadvantages, some of which are the following:

- XGBoost doesn't work so well on sparse data, and very dispersed data can create some issues.
- It has a black box nature: if we need effect sizes, XGBoost won't give us them.

3.4.2.3 Implementation

To train our data using the XGBoost model, we used the XGBoost library. We used the squared error for the objective function. To evaluate its performance, we used the mean absolute percentage error.

4 Results

In this section, we show the final results we obtained for both models, random forest and XGBoost. Despite the fact that we could not use the forecast data of meteo france models, both models outperformed the baseline model forecast of meteo france in validation score. Figure 4 shows that the Random Forest model is better than the XGBoost model as it achieved the score 0.44.

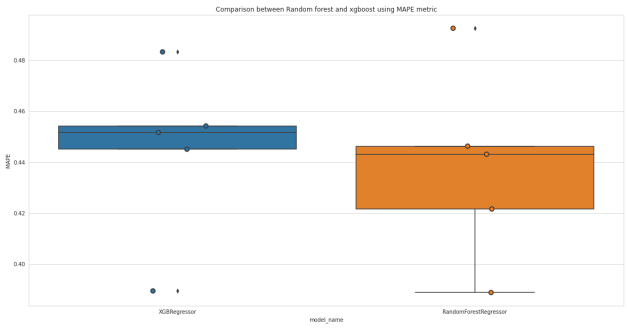


Figure 4: The percentage of the missing values in the features

However, after tuning the hyper parameters using the framework Optuna, XGBoost model performed better with a score of 0.36 as shown in Figure 5.

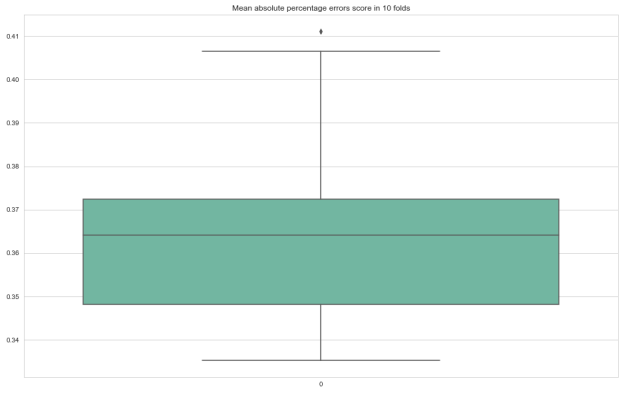


Figure 5: The percentage of the missing values in the features

References

[1] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

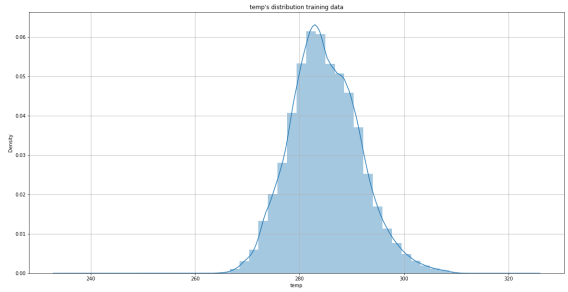
[3] Gláucia Tatiana Ferrari and Vitor Ozaki. Missing data imputation of climate datasets: Implications to modeling extreme drought events. *Revista Brasileira de Meteorologia*, 29:21–28, 2014.

[4] Min-Ki Lee, Seung-Hyun Moon, Yourim Yoon, Yong-Hyuk Kim, and Byung-Ro Moon. Detecting anomalies in meteorological data using support vector regression. *Advances in Meteorology*, 2018, 2018.

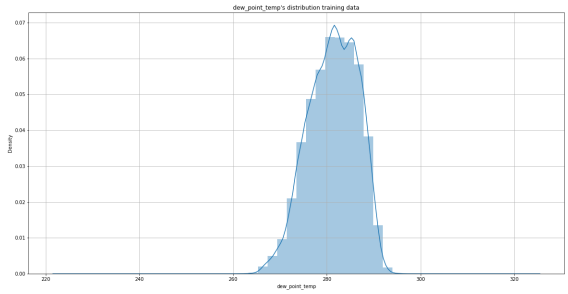
[5] Sungjae Lee, Yung-Seop Lee, and Youngdoo Son. Forecasting daily temperatures with different time interval data using deep neural networks. *Applied Sciences*, 10(5):1609, 2020.

A Data Exploration

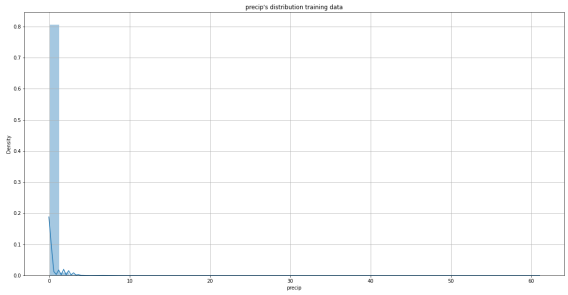
A.1 Feature's Distributions



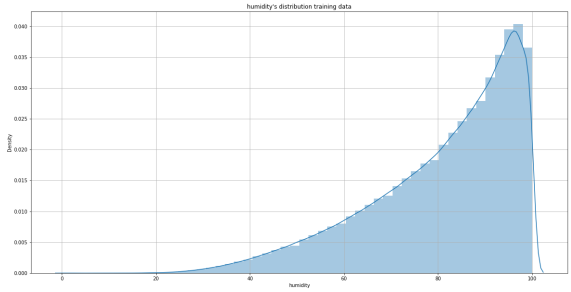
(a) Temperature



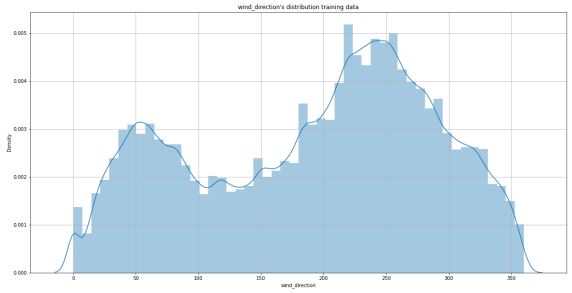
(b) Dew point temperature



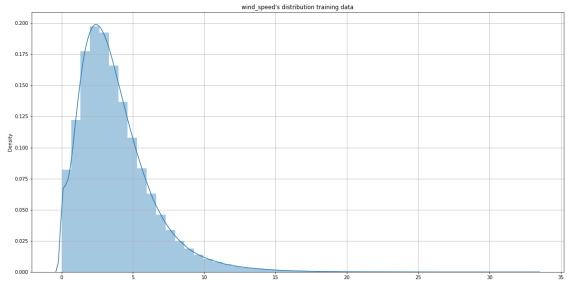
(a) Precipitation



(b) Humidity

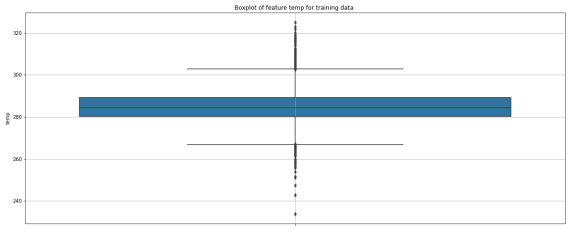


(a) Wind Direction

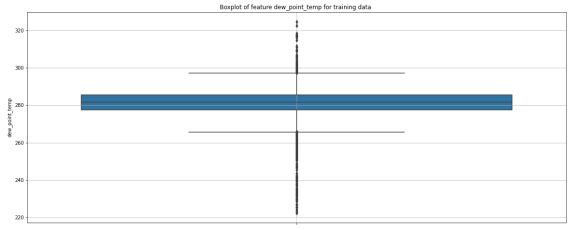


(b) Wind Speed

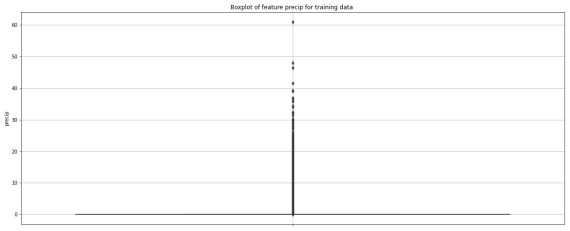
A.2 Box Plots



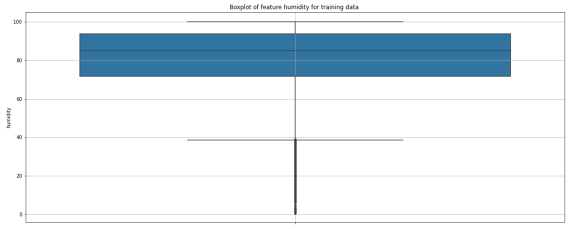
(a) Temperature



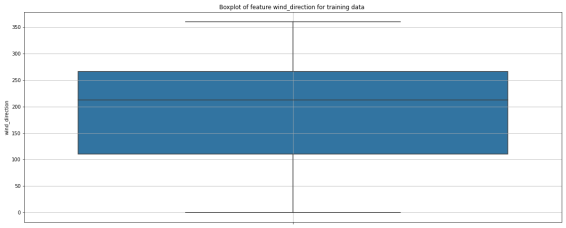
(b) Dew point temperature



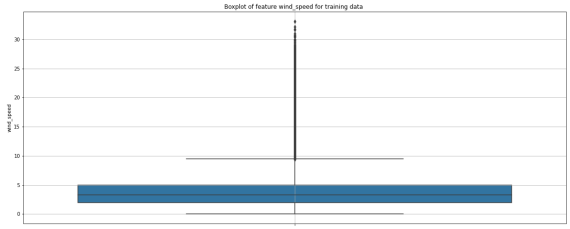
(a) Precipitation



(b) Humidity

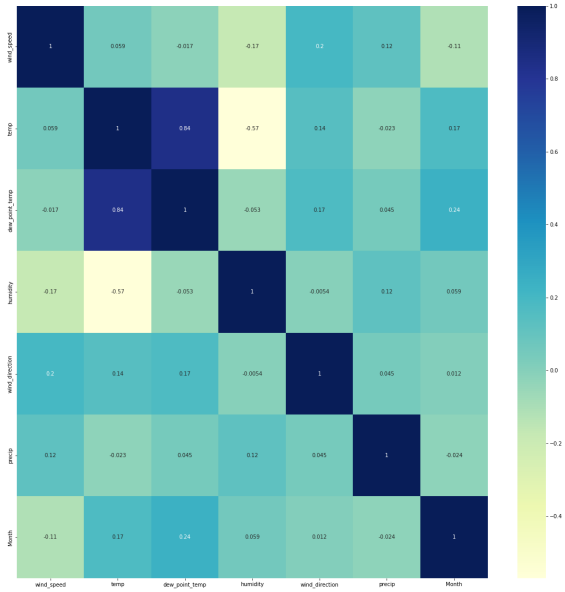


(a) Wind Direction

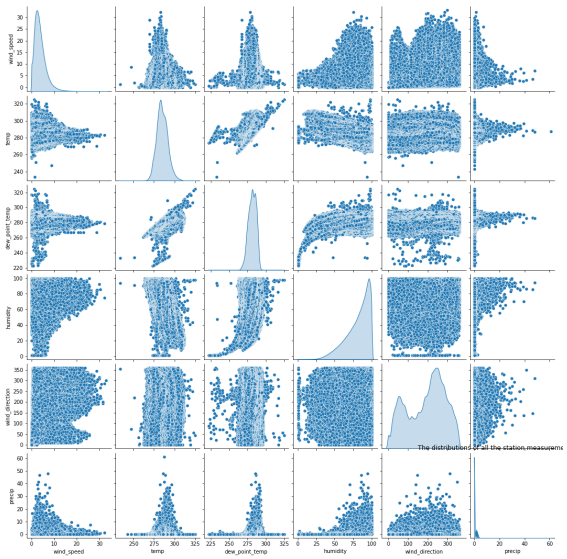


(b) Wind Speed

A.3 Linear Relationships



(a) Correlation Matrix



(b) Pair Plot

A.4 Non-linear Relationships

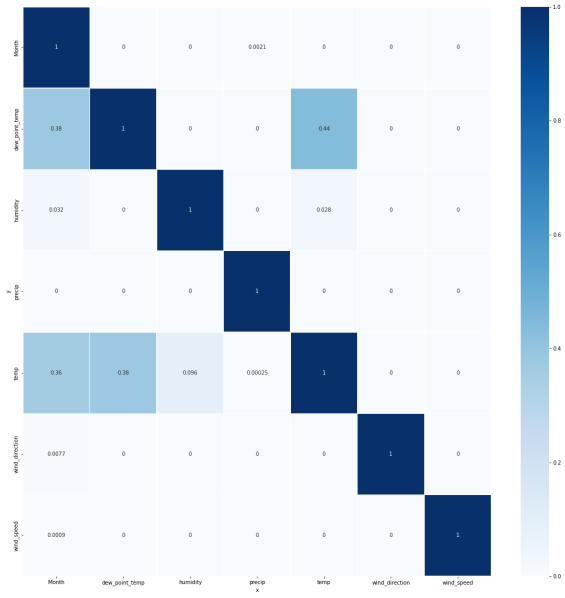
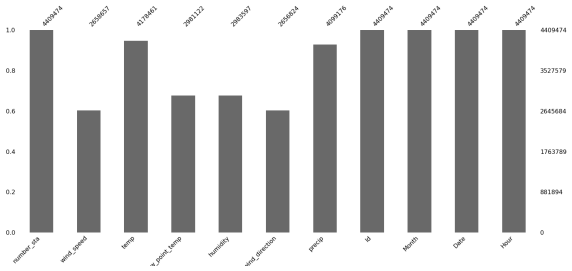


Figure 13: PPscore

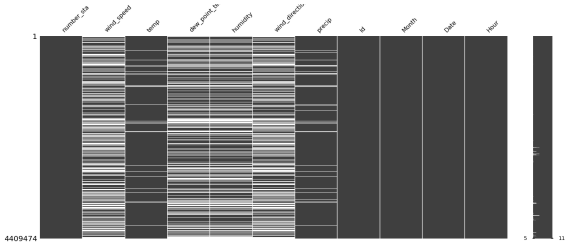
B Data Pre-processing

B.1 Missing Values

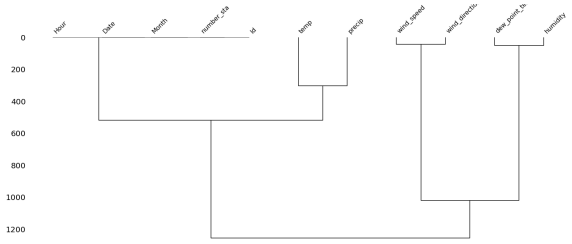
B.1.1 Exploration



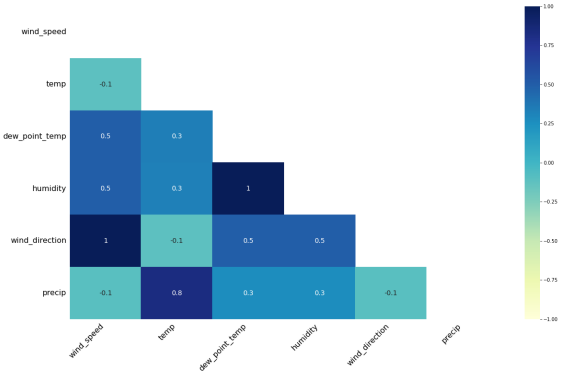
(a) Distribution



(b) Nullity Matrix

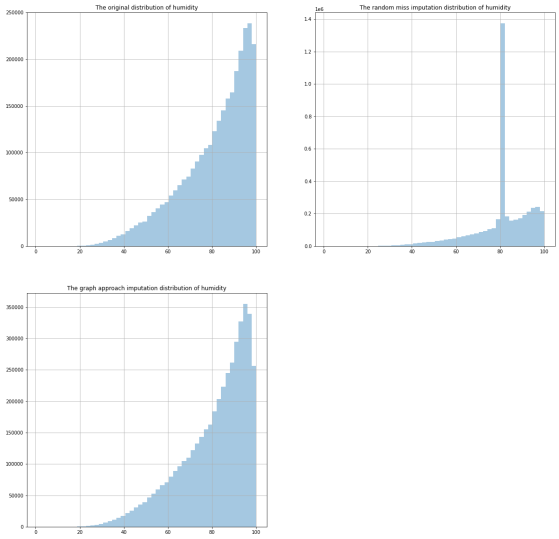


(a) Endro-gram

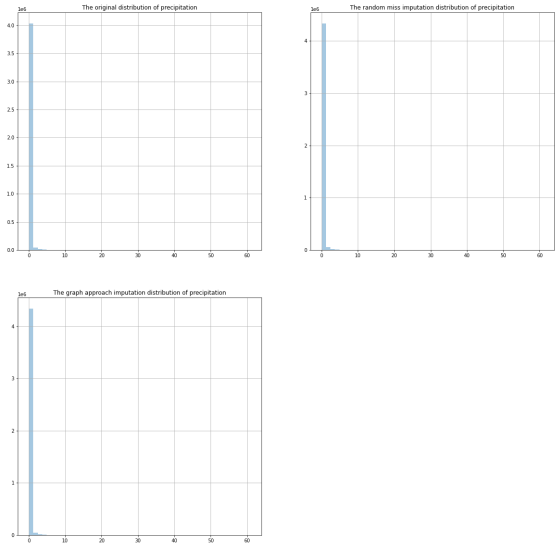


(b) Heat-map

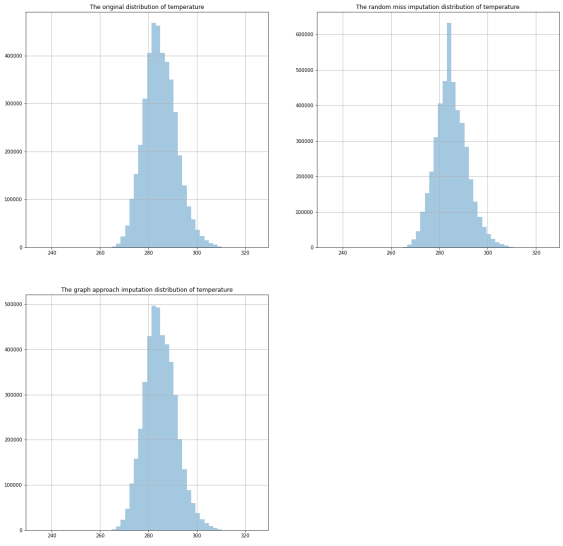
B.1.2 MissForest versus Graph Modelling Results



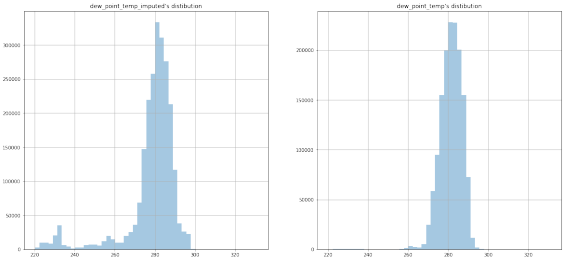
(a) Humidity



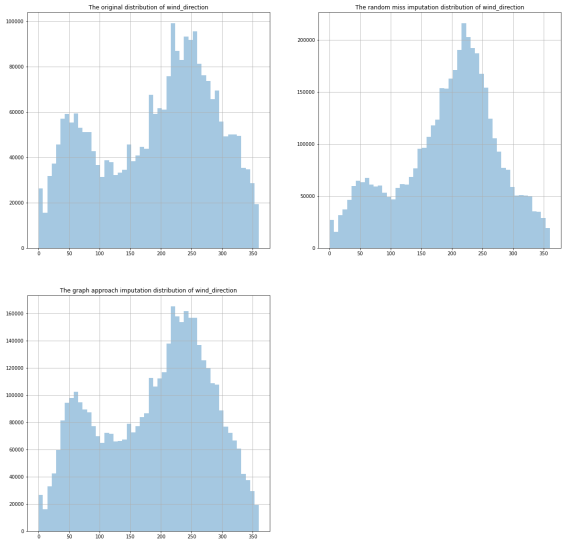
(b) Precipitation



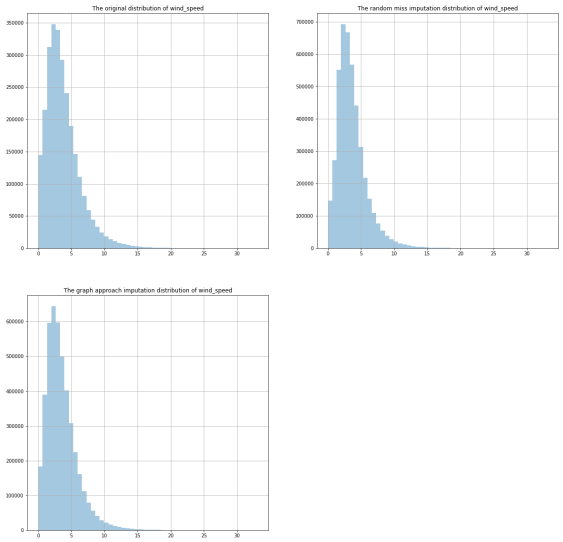
(a) Temperature



(b) Dew Point Temperature



(a) Wind Direction



(b) Wind Speed