AML Challenge 1: Weather Prediction
Group 47

Jana JUROŠ Jana.Juros@eurecom.fr
Rea PINTAR Rea.Pintar@eurecom.fr
Ivona STOJANOVIĆ Ivona.Stojanovic@eurecom.fr

February 14, 2024

1 Problem setting

The objective of this challenge is to provide a prediction of the temperature at different locations around the world and at different points in time.

2 Data analysis

A basic overview of the dataset will not help us in the future when we are doing data preprocessing (and ultimately model selection and training), as there is a large number of features, 111 in total to be exact. The data consists of various meteorological features that are dependent on the geographical location and the time they were taken. The features are highly heterogeneous, being of different types and scales.

Taking a look at the distribution of the train data in comparison to that of the test data, we can see that the test data covers some geographical locations that the train set doesn't include, or so-called *out-domain* data. At the point of doing model evaluation, we will see how this data will be affected and how accurate the predictions will be.

3 Data preprocessing

3.1 Handling missing data

The first step in data cleaning was replacing the missing values, which involved checking where specific features were available, and where there were *NaN* (Not a Number) or placeholder values. Since for every geographical location we have data coming from three different sources (GFS[1], CMC[2] and WRF[3]), we checked which rows didn't have available data from any of the sources using features containing the word *available*, so we might detect missing information more easily and subsequently handle the missing data. We also checked for other missing values in the dataset outside of availability scope, but there were none. Once we detected the missing data, we handled it using **Regression Imputation**. Regression Imputation is a process of estimating and filling in missing values relying on regression models to predict the missing values based on the observed data. In this approach, the missing data is treated as the dependent variable, while the remaining available data is used as independent variables or predictors. To conduct Regression Imputation, we performed the following steps:

- For each feature containing missing information:
  1. We split the dataset into two sets: one with complete observations (no missing values in any feature) and a set with incomplete observations (one or more features missing).
  2. We train a Linear Regression model to predict the chosen feature using the complete observations set.
  3. We use the trained model to predict the missing values of the chosen feature.
  4. We impute the predicted values in the dataset.

Regression Imputation is a popular method of handling missing data as it works for context-dependent features such as the ones we have here (all these meteorological features are highly dependent of one another, intuitively). There are other methods to deal with missing data, but Regression Imputation seemed like the

[1]Global Forecast System (GFS)
[2]Canadian Meteorological Center (CMC)
[3]Weather Research and Forecasting (WRF)

most suitable one. We can shortly present other methods and their flaws:

- **Deleting rows containing missing values from the dataset**: This way we lose data and possibly introduce bias if the missing values are not missing completely at random. If the probability of value missing depends on other variables or the value itself, deleting rows with missing values may result in a biased sample that does not accurately represent the population. This can lead to biased parameter estimates and inaccurate conclusions.
- **Imputation with a common value (empirical mean of the feature)**: Imputing missing values by setting them to the mean can introduce bias when the missing values are not missing at random - e.g. if the missing values depend on the value of another variable. In these cases, imputing missing values with the mean can distort the true underlying distribution and relationships in the data.
- **Hot deck imputation**: It is a form of imputation that attempts to preserve the relationships and patterns observed in the data. The idea is to fill in missing values in a dataset by borrowing values from similar or "neighboring" cases. Unfortunately, in order to find the nearest neighbors we would have to define distance between two samples taking into account as many features as we can. Given the different scales and distributions of all the features, we found it more practical to use Regression Imputation. This method considers all the features without requiring additional preprocessing steps.

We therefore chose Regression Imputation for several reasons:

- It attempts to **capture the dependencies and patterns** observed in the data. This can result in imputed values that align well with the existing relationships and patterns, thereby preserving the structure of the dataset.
- It **utilizes the information available in the dataset** (all samples and features) allowing for a more informed estimation compared to methods that rely solely on summary statistics or global measures.
- It can help **reduce potential biases** introduced by simpler imputation methods. It takes into account the variability and dependencies observed in the data, which can lead to more accurate imputations.

Having filled all the missing values, we no longer have need for the indicator columns (`cmc_available`, `wrf_available` etc.) and thus we are removing them from the set.

3.2 Handling outliers

We analyzed the outliers in the following way:

- For every feature of the dataset:
  1. We computed the outliers using the IQR (Interquartile Range) Method: we calculated the interquartile range by finding the difference between the third quartile (75th percentile) and the first quartile (25th percentile). Any value that fell below the first quartile minus a multiple of the IQR (in our case 1.5 times the IQR) or above the third quartile plus a multiple of the IQR is considered an outlier.
  2. We filtered all the features containing one or more outliers.
  3. We plotted a box plot for each of those features.

This way we were able to see the extreme nature of the domain. Outliers in this case are not misinformation, but rather the true state of the domain we are dealing with. That is why in weather prediction, it is often not advisable to remove outliers. Outliers in weather data can be valuable and contain important information about rare or extreme events that are of interest in weather forecasting and analysis. Removing outliers may lead to the loss of crucial data points that could affect the accuracy and reliability of predictions. By keeping outliers in the dataset, we can capture the full range of weather patterns, including the rare and extreme events that are critical for accurate modeling and forecasting. Hence, no outliers were removed after the analysis was conducted. This is especially reasonable, if we consider the out-domain test observations we have at hand.

3.3 Feature selection

The feature selection process, where we choose which features we keep and which ones we get rid of for lack of information, will be done in **two parts**. The first part is **removing (quasi) constant features**, i.e. variables that have little to no diversity, meaning they have a relatively low variance. To detect the presence of this kind of feature, we will set a variance threshold at 0.02 and classify any feature with a variance lower than this threshold as quasi-constant, and justifiably remove them as they don't bring any relevant information about the target feature. This threshold has been derived by taking a closer look at a histogram of all the variances from the variables in the training set (after having scaled the data). This threshold helps us differentiate lower-range outliers from other variances which ensures that we're only focusing on non-constant features that have predictive power. Before applying the threshold it is necessary to scale the data, as we have features that are contained on vastly different scales, and the implemented function we are using, `VarianceThreshold`, does not take this into account. For scaling we are using `MinMaxScaler`, as it scales the data to a fixed range [0, 10], making the variances comparable to one another. Using `StandardScaler` would have defeated the purpose of using a variance threshold, as that type of scaling makes the variance equal to 1. Using the described method, we eliminated 19 (quasi)constant features from the set.

The second part of feature selection is **removing highly correlated features** which we will be doing by checking the correlation coefficients between the features. Of course, we want to check which feature is most highly correlated with the target value `fact_temperature` (in our case, it was the feature named `gfs_temperature_97500`) and keep it, as it is our most reliable predictor of the target value in regression models. However, we want to remove features that are highly correlated to that feature or others. By visual inspection of the correlation matrix, we set a threshold of 0.84 and removed any features that have a correlation coefficient higher than the threshold.

3.4 Recursive Feature Elimination

Finally, we use **Recursive Feature Elimination (RFE)** to choose the best set of features for a given task. RFE is a feature selection technique that selects a subset of features by recursively removing the least important feature, until a desired number of features is reached. At each iteration, a model is trained on the remaining features and the feature with the lowest importance score is removed. One of the advantages of RFE is that it can be used with any machine learning algorithm that provides feature importance scores.

In our case, we experimented with training the model using different numbers of features selected by RFE. We trained the model using 5, 10, 15, and all features, and observed the performance. After comparing the results, we found that the best performance was achieved when using all features. Therefore, we present the results below based on the model trained with all features.

4 Model selection

With the type of variable our target feature is (`fact_temperature` - a continuous variable signifying the temperature in Celsius), it makes sense to proceed using regression models as they are often used to predict the outcome of a continuous variable. In our case, we will be trying out 2 models - Bayesian Linear Regression and Gradient Boosting Regression. For both models, we will be using the data given in `train.csv`, which we will be splitting into a training set and a validation set using the 80%-20% ratio, which is commonly used as it balances the need for a sufficiently large training set to accurately learn the model with the need for a large enough validation set to provide a reliable estimate of the model's performance.

4.1 Bayesian Ridge Linear Regression

This model is a form of Linear Regression which is fundamentally different from the classical approach. It is not based on point estimation of the coefficients, but rather on determining a distribution of those coefficients, as well as allowing regularization of the coefficients. This means that it is a quite flexible approach which is good at avoiding overfitting. Additionally, we hope to accommodate the out-domain test data in that way, since one of the strengths of this model is generalization. The parameters we will be optimizing for this model include $\alpha_1, \alpha_2, \lambda_1, \lambda_2$ which tell us about the shape of the prior as well as the regularization. In all cases, higher values of these parameters correlate to higher regularization.

4.2 Gradient Boosting Regression - XGBoost

Gradient Boosting Regression, more specifically the XGBoost version of the model, is an ensemble technique which combines multiple decision trees. It is built in an iterative manner: In each step a decision tree is fitted and the residuals are computed. For the subsequent step the decision tree is also trained on these residuals, so that this decision tree corrects the errors from the previous ones. This approach is very versatile as it is able to also capture more complex, non-linear relationships, and is generally a powerful ensemble learning method that combines the strengths of multiple weak learners to achieve high predictive accuracy.

5 Parameter optimization

As both of our models have a large number of parameters we could optimize, **Grid Search** seems to be a good approach to optimize parameters by finding the best combination. For both models we will optimize the parameters using Grid Search and in the case of Bayesian Ridge Linear Regression we will be using it with 5-fold **Cross-validation**. This step of Cross-validation has been left out optimizing the parameter of the XGBoost model because of time constraints (runtime of over 6 hours).

6 Model evaluation

To evaluate how well the model performs, we chose the Root Mean Squared Error (RMSE), a metric widely used for regression problems. We also took care to rescale the response prior to calculation to obtain interpretable results.

With Bayesian Ridge Linear Regression, we optimized four parameters: $\alpha_1, \alpha_2, \lambda_1, \lambda_2$ which tell us about the prior as well as the regularization of the coefficients. For each of these we tried 3 different values: $1e^{-6}, 1e^{-5}, 1e^{-4}$, with $1e^{-6}$ being the default for all four parameters. The optimal values of those parameters (the combination that yielded the lowest RMSE) is the following:

$$\alpha_1 = 1e^{-4}, \alpha_2 = 1e^{-5}, \lambda_1 = 1e^{-6}, \lambda_2 = 1e^{-4}$$

So we can see that the optimal values are mostly different from the default, with the exception of λ_1. Below is the comparison of RMSE of the basic model (all default parameter values) and the model after Parameter Optimization (P.O.).

Bayesian Ridge Linear Regression	RMSE on basic model	RMSE after P.O.
Train set	2.6333	2.6333
Test set	2.6345	2.6345

The Parameter Optimization didn't affect the RMSEs by a lot. There were only minor changes in the trailing decimal places.

As for the XGBoost, the parameters we optimized were the learning rate (with choice of $\eta = [0.1, 0.001]$) and the regularization term (with choice of $\lambda = [0, 0.01]$. We also reset the parameter max_depth to 4 rather than its default value of 6 (both in the simple model and in the Grid Search) for two reasons. The first reason being that lower depth of the tree might decrease overfitting, and the second reason being the length of executing the Grid Search (lack of resources). For the same reason, we chose not to try out more combinations or optimized a larger number of parameters, as it was quite computationally expensive. The resulting optimal parameters for XGBoost are:

$$\eta = 0.1, \lambda = 0.01$$

These are actually close to the default, except for λ. Since the value is not zero, it means that regularization is taking place. Below are the results comparing the RMSE of the basic model and the model after P.O. for XGBoost.

XGBoost model	RMSE on basic model	RMSE after P.O.
Train set	2.2378	2.3361
Test set	2.2450	2.3415

7 Conclusion

In the end, we decided to go with the XGBoost Model for which we tuned the parameters, since it yielded the best results during our model training. As a final score our model reached 2.1227 on Kaggle. The results we presented here are quite satisfactory considering that the baseline model (that's provided in the Kaggle competition) only managed to reach a RMSE of about 8.48.