AML - Weather Prediction - Challenge 1
Group 4

Philipp Peron
philipp.peron@eurecom.fr

Khristen Thornburg
khristen.thornburg@eurecom.fr

Hugo Rechatin
hugo.rechatin@eurecom.fr

Ioannis Panagiotis Pitsiorlas
ioannis.pitsiorlas@eurecom.fr

1. Introduction

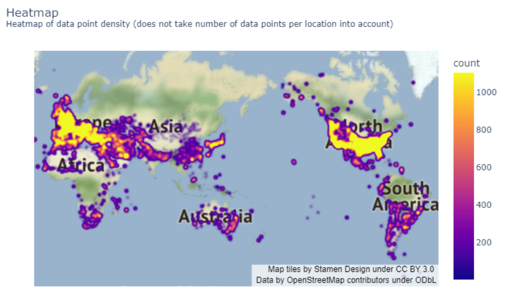
Dating back to as early as the classical era, the art and science of weather hasfascinated man. From predicting the winds to planning sailing voyages to predicting rainy seasons for hunting, weather forecasting has been essential to daily life. While the goal of predicting the temperature using historical data remains essentially the same, the tools we use today have greatly improved the quantity and quality of the data that can be process and analyzed in order to make much more accurate predictions. In this lab, we consider six months of weather data with 111 features collected across 8,053 different locations around the globe in an attempt to predict the temperature for unseen data. We begin by analyzing the data to understand how the features are related and which ones are most important in weather prediction. Next, we move to data engineering, where we perform data normalization and feature selection in order to simplify the analysis. We also consider how to deal with null values in the data. Finally, we come to model selection where we primarily considered three different models: XGBoost, a Random Forest and CastBoost. In the end, the model with the best performance was CastBoost which gave us an RMSE-score on the preliminary test set of 1.62.

2. Data Analysis

Because of the large number of features and samples (111 and nearly 2 million, respectively), visualizing the data with various graphs was essential in understanding the dataset, performing feature selection and choosing a model that uses the data logically.

The heatmap in figure 1 shows the density of the locations with available data. From this, we can see that the majority of the monitored locations are in the Northern Hemisphere and that there is little data in the training set around the equator, near the poles and in the Southern Hemisphere. This could potentially be problematic because the temperatures behave differently depending on the locations. For example, temperatures around the equator vary much less with the change of seasons.

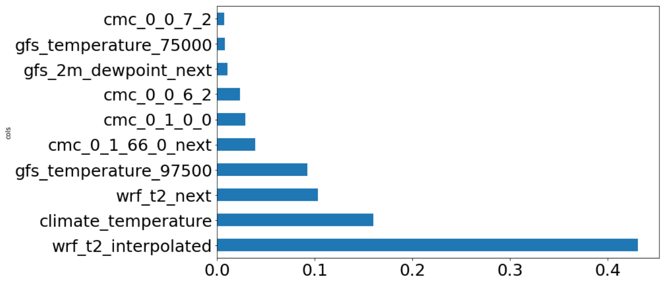
Figure 1. Datapoints per location


Heatmap
Heatmap of data point density (does not take number of data points per location into account)
count
1000
800
600
400
200
Asia
Africa
North America
South America
Australia
Map tiles by Stamen Design under CC BY 3.0
Data by OpenStreetMap contributors under ODbL

2.1. Feature Importance

To get an indication of which features might be more useful than others in predicting the temperature, we looked at how each feature correlates with the temperature. Figure 2 shows the 10 features with the strongest correlation to the temperature. The feature "wrf_t2_interpolated" has the highest correlation with 0.96.

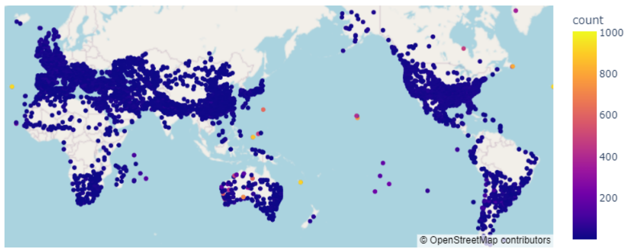
Figure 2. 10 most target-correlated features (offset=0.5)



2.2. Null values

Another important consideration when training a model is how null values are treated. In our training dataset, about 5% of the samples included a NaN value. The map in Fig. 3 shows that the null values are evenly distributed among the locations in the training set. The majority of locations have less than 10 null values while only about 15 locations have 500 to 1000 null values.

Figure 3. Number of null values by location



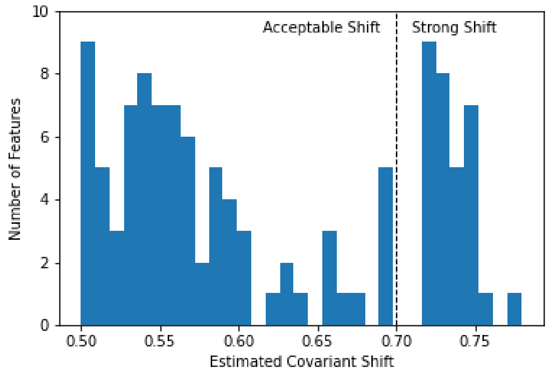
3. Data Engineering

We started data engineering by doing simple data normalization to improve the data processing thereafter. To make the preprocessing easily tunable we created a DataPreprocessing class that took a config-dictionary as input and included all our preprocessing methods.

3.1. Covariant shift

Most of the test data measurements are taken from different locations than the training data. This likely leads to a strong covariant shift from the training data to the test data. Therefore, we tried to estimate the covariant shift for the different features. For each feature, we trained a random forest classifier (RFC) [2] to predict whether a sample was from the training set or the test set. We used the RFCs' accuracies to assign each feature a score indicating the covariant shift. The better the RFC can classify from which data-set a sample is with a single feature (high accuracy), the more likely a strong covariant shift is in that particular feature. Figure 4 shows the distribution of the covariant-shift-scores. From the 111 features, 31 had a score of more than 0.7. These features likely cause problems during inference on the test data set. To reduce the potential problems, we removed most of the features with a shift high score, still keeping those features that we found to be very important during data analysis.

Figure 4. Covariant shift estimated by a random forest



3.2. Additional Data

Two features that we thought could be helpful to add were the elevation and the hour of the day. By using the latitude and longitude data, we got the elevation at each location from the public SRTM3 elevation model [1].
To find the exact hour of the day the reading was taken, we would need to take look up the time zone of every location. Instead, we simplified the task by just considering the time zone offset, which can be approximated based on the longitude. Greenwich is longitude 0 and time zone UTC and longitude of +-180 degrees is 12 hours of time offset with linear interpolation in between.
Additionally, the four most important features were squared and added to the dataframe.
We further extended our data set by adding the top 10 principal components which were found via a PCA.
Overall, our training set for the final model included 128 different features.

3.3. Dealing with Null Values

When we first trained the models, we simply deleted the samples that contained null values. While this was an easy way to deal with missing values, it caused us to lose some important information. In order to attempt to retain the maximum amount of data, we set the null values to the average value of the feature.

3.4. Removing Outliers

When certain features did not have data for a location, the fields were null. However, some features did not have readings for a specific location, but still had a value saved. We saw this in the gfs_soil_temperature feature. Instead of containing null values when there was no data, this field had a value of -9999 °C. We simply removed these values from the dataframe to avoid having them skew the results.

3.5. Scaling

Machine learning algorithms are often sensitive to skewed distributions. That is why we tried different methods to scale the data. After trying min-max-scaling and standard-scaling, we settled on the robust scalar. The robust scalar centers the data around zero and scales the quantiles individually. This makes the scaling method less sensitive to outliers.

3.6. Binning

In an attempt to change our approach to the regression exercise, we turned to binning. To do so, we sorted the base dataframe by increasing fact_temperature and then created N classes containing equal amounts of samples (class 0 containing the lowest temperatures and class N-1 the highest). We then trained an XGBoost regressor to classify our samples according to these bins. After testing with 10, 20, 30 and 60 bins we achieved the best result for 20 bins. Yet we did not include it in the final CatBoost model since it decreased the performance overall.

4. Model Selection and Hyperparameter Optimization

4.1. Hyperparameter Optimization

An important part of creating a model that describes the data well without overfitting is the tuning of the hyperparameters. Since we only worked on this project for two weeks, we were not able to explore all possible options to best optimize each model, but we describe in each section below which hyperparameters we altered to attempt to fine-tune the models.

4.2. XGBoost

The first model that we tried was XGBoost. Extreme Gradient Boosting is a tree-based boosting algorithm that is known to use more accurate approximations in order to find the best tree model [3]. The parameters that we used for our baseline model were a learning rate of 0.03, a max depth of 4, an alpha of 6 and the number of estimators equal to 300. Using these parameters and the binning technique described above gave us an RMSE of 2.09 on the preliminary test set and took about 35 minutes to train.

4.3. Neural Network

Dense Neural Networks are commonly used in tabular Regression and can handle large feature vectors. We attempted to train a neural network, but after 10 epochs it had a very bad performance, so we decided to not pursue this route.

4.4. Random Forest

Our second most successful model was a random forest regressor (RFR). The best RFR achieved an RMSE of 1.96 on the preliminary test set and used: 120 estimators, a maximum tree depth of 25, a 3 minimum sample count before a split and 2 minimum samples per leaf. We first did basic graduate tuning to get an int for the hyperparameters in the context of the data. After that, we used that knowledge to define the values for a grid search. For the grid search, we used a slightly smaller subset of all the data. In the end, we retrained the classifier on the entire labeled data set with the optimal hyperparameters.

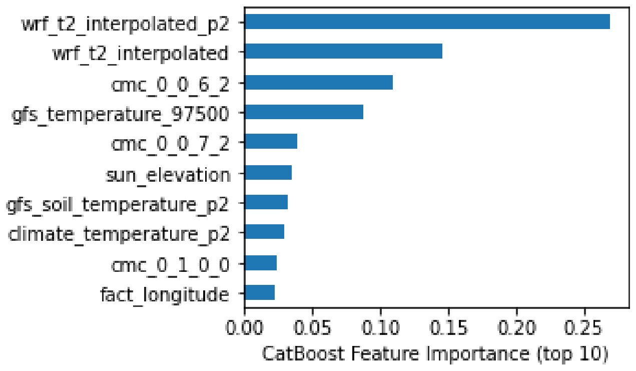
4.5. Ensemble

To combine the expressiveness of different models, we also experimented with an ensemble model composed of 4 individual models: Gradient Boosting, Random Forest Regressor, XGBoost and Ridge Regressor. Hereby, we applied weights to the individual models for the overall prediction based on their scores on our validation set. Unfortunately, this ensemble regressor did not perform any better than the random forest regressor on its own and took almost 12 hours to train.

4.6. CatBoost

Our most successful model overall on the validation data (and the preliminary test data) was a CatBoost model which uses gradient boosting on decision trees [4]. For the hyperparameter tuning, we performed a grid search over several combinations of the number of iterations, the depth of the trees and the learning rate. The optimal parameters that we concluded on were: 15,000 iterations, a tree depth of 8 and a learning rate of 0.1 which yielded an RMSE of 1.64 on the preliminary test data.
Figure 5 shows the top 10 most important features. Hereby 'wrf_t2_interpolated_p2' is the most important feature, followed by 'wrf_t2_interpolated'.

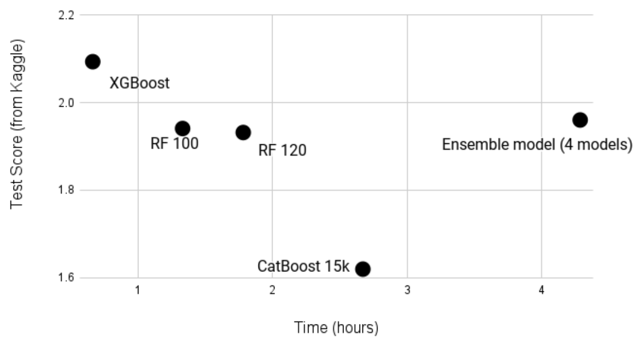
Figure 5. CatBoost feature importance of top 10 features



5. Conclusion

Overall, we pre-processed the data by: imputing null values, estimating the covariant shift and removing the features with a large drift, adding new features like elevation, principal components from a PCA and the hour of the day. Following that we tried out different models like neural networks, XGBoost, Random Forest, Ensemble models and CatBoost. Figure 6 shows the overall score vs the amount of time it took each model to train. The CatBoost model was by far the fastest, taking only 3 minutes to run and while giving us a much better score.

Figure 6. Test Score vs Time to run for each model



References

[1] SRTM3 kernel description. https://lpdaac.usgs.gov/products/srtmgl3v003/. Accessed: 2022-05-5. 2

[2] Leo Breiman. Random forests. 2001. 2

[3] Tianqi Chen and Carlos Guestrin. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016. 3

[4] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features, 2019. 3