Smarter Mobility Data Challenge

Daniel Hebenstreit, Thomas Wedenig

1 Data Exploration & Pre/Post-processing

Let $\mathcal{S}$ denote the set of stations and $\mathcal{T}$ denote the set of targets with $|\mathcal{S}| = 91, |\mathcal{T}| = 4$.

Our exploratory experiments did not show signs of a trend within the series. Regarding stationarity, we performed the Augmented Dickey–Fuller test on the daily averages of target values for each station and found inconclusive results: We cannot assume stationarity for all target-station pairs in $\mathcal{S} \times \mathcal{T}$, which is why we employ differencing in the construction of our ARIMA model (Sec. 4). We hypothesize that we cannot reliably forecast the high-frequency fluctuations of the target values multiple weeks into the future. We call this the *simple model hypothesis* and hence, we focus on simple models that capture and predict the expected target values. To enforce this behavior, we preprocess the data by computing a rolling window average of the time series. We empirically evaluate different window sizes and find that a window size of 2.5 hours works best.

During our data exploration, we encounter a significant change in the behavior of the individual stations after COVID regulations were enforced in Paris. The approximate date of this behavioral changing event is October 20, 2020. We perform several empirical tests which show that excluding the data improves the performance of our models, which is why we restrict the training data to values that were recorded after the specified date.

We also add custom features, namely a column that displays if the current date is a French holiday, as well as sine and cosine transforms of *time of day*, *day of week*, *month*, and *day of year*.

After the COVID restriction were enforced, we assume that several stations were turned off, as labels were missing over large intervals. Thus, we experiment with different methods of missing value imputation, including mean value imputation, spline interpolation, back-filling, and forward-filling. However, none of these approaches perform better than just neglecting the missing values and dropping the timesteps with missing values, which is therefore the approach we choose.

Furthermore, our regression models often return non-integer outputs which often violate the restriction that the predictions for a single station needs to add up to 3 for each timestep (as stations only have 3 plugs). We try a number of post-processing methods to enforce this constraint, such as rounding and rescaling the outputs.

2 Tree-based Regression Model

Using `skforecast`, we train an autoregressive XGBoost model with 100 estimators. We trained it on all of the 91 stations individually, each having 4 targets, resulting in $|\mathcal{S}| \cdot |\mathcal{T}| = 364$ models. Each model receives the last 20 target values, as well as the sine/cosine transformed time information as input and predicts the next target value. We also discarded all features that are constant per station, for example, station name, longitude, and latitude. The final regression model achieves a public leaderboard score of 177.67.

3 Tree-based Classification Model

To overcome the issue of normalized outputs (i.e., that they sum to 3), we transform the regression problem discussed above into a *classification* problem: Consider the set of possible target values for a given station and a given timestamp:

$$\mathcal{C} = \left\{ \mathbf{x} \in \{0, 1, 2, 3\}^4 \quad \text{s.t.} \sum_{i=1}^{4} x_i = 3 \right\}.$$

We now solely consider an integer index i_c for each element $c \in \mathcal{C}$ (by arbitrarily enumerating all elements). Since $|\mathcal{C}| = 20$, the set of indices becomes $\mathcal{I} = \{0, \ldots, 19\}$. We note that $\mathcal{I}$ loses the ordinal information present in $\mathcal{C}$, which is clearly a drawback of this naïve approach. However, this approach performed really well. We trained a single XGBoost classifier with 300 estimators for all stations, achieving a public leaderboard score of 178.9. We also tried to include the results of previous timesteps as features, as the auto-regression part (using information of previous timesteps) also improved our XGBoost regression model. However, this did not achieve the expected results in the classification approach.

4 ARIMA Model

Under the assumption that simple models generally perform well on this prediction task, we fit a non-seasonal autoregressive integrated moving average (ARIMA) model for each combination in $\mathcal{S} \times \mathcal{T}$. To predict the value of a particular target, we only consider the last p values of the same target (in the preprocessed time series) and do not utilize any exogenous variables for prediction (e.g., time information).

An ARIMA model of order (p, d, q) is defined as

$$\left(1 - \sum_{i=1}^{p} \alpha_i L^i\right) y'_t = \left(1 + \sum_{i=1}^{q} \alpha_i L^i\right) \varepsilon_t$$

where L is the lag (or backshift) operator, y'_t is the d-order differenced time series data, α_i are the parameters of the p-order autoregressive part of the model (AR), θ_i are the parameters of the q-order moving average part of the model (MA), and ε_t is assumed to be i.i.d. white noise.

In particular, we choose $p = 2$, $d = 1$, $q = 1$ for all models we fit since we have empirically experienced good performance on both the validation set and the public leaderboard. We have also tried estimating p, d, q using `auto_arima`[1], although this has not improved performance on the validation set.

We observe that the forecasted values using these models have very low variance, i.e., each model outputs an approximately constant time series. Surprisingly, these predictions already achieve a competitive score on the public leaderboard (third place with a score of 177.09), which is further evidence for our *simple model hypothesis*.

5 Ensemble

The final model is an ensemble of the tree-based regression model (Sec. 2), the tree-based classification model (Sec. 3), and the ARIMA model (Sec. 4): For a single target, we compute the weighted average of the individual model's predictions (per timestamp). The ensemble weights were chosen to be proportional to the public leaderboard score, namely $w_{\text{reg}} = 0.35$, $w_{\text{class}} = 0.25$, $w_{\text{ARIMA}} = 0.4$. The large weight for the ARIMA model also encourages our *simple model hypothesis*.

Since the predictions of the tree-based models have high variance, we can interpret mixing in the ARIMA model's predictions as a *learned regularizer*, which decreases the variance of the final model. As the tree-based models also use time information for their predictions, we utilize the entirety of the available features.

6 Conclusion

We train three simple, interpretable models with different inputs on different scopes of the data (e.g., per station and target, or for all stations and targets) that all feature a different bias-variance trade-off. All models can be trained efficiently on a single machine and we ensemble their post-processed outputs to obtain robust predictions. We further conclude that it is practically impossible to reliably predict high-frequency human behavior (which we thus model as random noise) over 19 days and that simple models empirically outperform all complex *Deep Learning* approaches we evaluate on this task, including neural networks, a Transformer architecture, and a LSTM network.

[1] `https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html`