Predicting water levels of Lake Constance

Case description

We are predicting the next-day water levels of Lake Constance using a comprehensive set of features, including daily weather characteristics such as humidity, mean temperature, precipitation, snow depth, sea level pressure, sunshine duration, global radiation, and wind speed, but also moon cycles, groundwater levels, and climate change.

During periods of high water levels, the steeper entry and exit angles on ferry ramps can make boarding and disembarking difficult for vehicles, especially low-profile ones, which may even be restricted to prevent potential damage. This situation often requires drivers to proceed with extra caution, increasing the risk of delays. Conversely, low water levels also present challenges, such as reduced cargo capacity due to loading restrictions and potential delays in operations. Additionally, inefficient load distribution under these conditions can lead to higher fuel consumption. These operational difficulties contribute to customer dissatisfaction, revenue losses from unoptimized operations, and increased overall operational costs, creating a persistent challenge for ferry companies in the region.

Our predictions will allow ferry companies operating Lake Constance to be able to plan operations for the next day with high confidence of water levels and be able to announce issues and suspend operations early. Ultimately, it will ensure smoother business operations and enhanced certainty for all stakeholders.

Feature selection

The primary focus of the model is to predict the **water levels of Lake Constance**, making it the target variable. Its long historical data (1924-now) provides a large basis for detecting patterns and trends over time.

Weather data includes several **daily variables** which highly correlate to water levels. For instance, **humidity** levels influence evaporation rates. Higher humidity generally leads to lower evaporation, which helps maintain water levels, while low humidity increases evaporation. Then, the **mean temperature** impacts both evaporation and snowmelt. Warmer temperatures increase evaporation and lead to earlier snowmelt. This contributes to seasonal water level fluctuations. Also, the daily **precipitation** is a direct contributor to water inflow into the lake. Rainfall increases water levels by adding to surface runoff, tributary inflow, and groundwater recharge. Furthermore, the **snow depth** serves as a reservoir of water. When snow melts, it contributes to water inflow during specific periods. This causes seasonal fluctuations. **Atmospheric pressure** affects weather patterns such as storms or prolonged dry periods. Low sea level pressure can be associated with storms and precipitation, while high pressure often correlates with stable and dry conditions. **Sunshine duration and radiation** influence evaporation and indirectly impacts water temperature. The latter can further drive evaporation rates. Besides, wind enhances evaporation by increasing the surface area exposed to air. It can also cause short-term water level fluctuations due to wind waves or oscillations. Therefore, we take the **wind speed** as a feature.

Additionally, we selected a dataset about **moon illumination**. Moon cycles are insightful because lunar phases affect tidal movements and water levels. While Lake Constance is not directly tidal, moon cycles still influence local hydrological patterns. For example, subtle changes in groundwater and atmospheric pressure can occur depending on the moon state. Including this feature improves the model as it allows to test any correlation between moon phases and water level anomalies.

Groundwater also plays a role as it directly contributes to inflow of water into the lake. It can also act as a buffer during periods of low precipitation. Changes in groundwater levels can indicate shifts in regional hydrological systems. Including this feature complements surface water data. It adds a regional subsurface perspective to the hydrological cycle.

Finally, we selected **climate change** data. The latter reflects long-term trends in temperature, precipitation, and seasonal snowmelt patterns. With historical climate change data, one accounts for shifts in conditions such as variability in precipitation and in snowmelt timing.

Main Steps

1. Preprocessing

Preprocessing ensures the consistency of data used for building predictive models. We started by gathering data for our features from different sources. The water and groundwater levels are from the Baden-Württemberg State Institute for the Environment (LUBW). The measurements for the Lake Constance water levels are taken in Constance, while the groundwater levels are from a location nearby. Weather data is taken from the European Climate Assessment & Dataset project, where different weather sources are aggregated to fill missing values. Moon illumination data was exported from NASA's Horizons System, where the data is precisely calculated for a geocentric location. Climate change data was taken from the IMF as annual mean surface temperature change in Switzerland.

We converted all data to csv and removed unnecessary information from the files. Next, we merged the features based on the date and used a forward fill to repeat values from the last observation up to the next data point. Thus, we filled the missing values in the weekly groundwater levels and annual climate change observations.

Since we aim to predict tomorrow's water level using today's weather data, the target needed to be shifted back by one row relative to the features. This ensures that each row contains the features from a given day and the corresponding target for the following day. Without this shift, the model would incorrectly associate today's features with today's water level, leading to misaligned predictions.

Outliers were identified through time-series plots. Through this, deviations in features were flagged such as unusual pressure values on June 2nd, 2023, or excessive wind speed values on February 12th-13th, 1994. These outliers were not immediately imputed, as we had to preserve the integrity of the dataset for later train-test separation. The identification of outliers allows to maintain natural variability and ensure less bias for future imputation strategies.

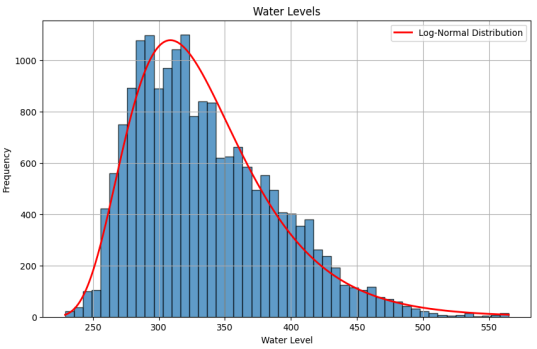
The dataset was divided into training and testing sets to make the evaluation of models easier. Data prior to 2018 was used as training data, while subsequent data was reserved for testing. This temporal split minimizes overfitting as the model is evaluated on unseen future data. Additionally, features such as lagged water levels were aligned across the split to prevent data leakage. With this separation, we preserved the integrity of the testing dataset.

We identified missing values through columns like snow depth, sunshine, wind speed, and global radiation. To address this, seasonal means were calculated for each month. Hence, imputation preserved seasonal patterns. For outlier corrections, we used rolling mean values within a 30-day window. We avoided data leakage by relying only on historical and contemporary data. Missing values were imputed separately with seasonal mean imputation for the training and testing datasets to prevent bias during model evaluation. The dataset was filtered to cover the timespan from 1973 to 2023. This provided a comprehensive temporal coverage for predictive modeling.

2. Feature Engineering

We added lagged features for the past 7 days as water levels often depend heavily on their recent history. Lagged features enable the model to understand how current values are influenced by the past. Thereby, we did not leak data because we assume that today's and previous days' data are available at the time of prediction.

Plotting the target value, we saw that the Water levels are right-skewed, and their distribution perfectly fits the log-normal curve. Skewness can exaggerate trends and seasonal patterns, making it difficult to identify and remove these components accurately. Furthermore, by applying the log transformation, large values in the dataset are compressed. This stabilization helps reduce the impact of outliers and brings the data closer to a normal distribution.

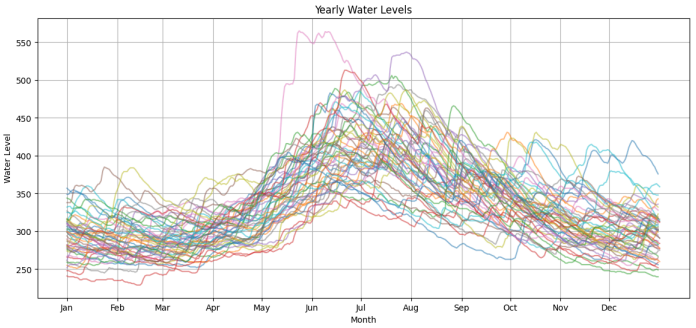


Through plotting, we saw clear seasonality (half-year and yearly peaks in the Fourier Transformation) of the water level data; a clear trend was not visible. To cope with the trend and seasonality of time series data, we chose two different approaches, as different modeling techniques require different preprocessing strategies depending on their assumptions and the type of relationships they aim to capture.

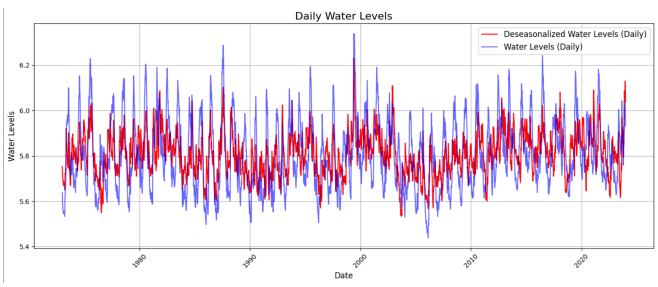
Approach 1: Retaining Trends and Seasonality with Features for Random Forest and Neural Networks

For Random Forest and Neural Networks, we did not explicitly remove trends and seasonality from the data, as the two models can learn these relationships directly from the data. Instead, cyclical features such as sine and cosine transformations of time-related variables (i.e., year, half a year, month, week) were added to the dataset. These cyclical features capture the periodic nature of the data without removing seasonal patterns. This approach avoids the potential loss of information from detrending or deseasonalizing. Furthermore, even though no trend was visible (statistically, see Approach 2) we also added a time index as it can help the models understand the sequence of events and detect subtle temporal patterns. As Neural Networks are sensitive to the scale of input features because weights are adjusted during backpropagation, standardization was performed on the features (excluding the target variable and cyclical features).

Approach 2: Removing Trends and Seasonality for ARMA and Linear Regression



For ARMA models and Linear Regression, trends and seasonality were removed through detrending and deseasonalization techniques. This step ensures that the data meets the stationarity assumption for ARMA models and the linear relationship assumption for regression. We began by examining seasonality in the features and target variable through a combination of visual and statistical analysis, plotting the yearly recurring patterns of the features and water levels and calculating with the correlation coefficient the linear relationship between their seasonal components to identify shared seasonality. For features showing moderate correlations (absolute value ≥ 0.5) with water levels, we compared their Fourier transformations (to reveal dominant periodicities) to those of the water levels. The target value was deseasonalized and features with shared seasonalities (i.e., their yearly and half-yearly patterns) too. This step was crucial because shared seasonalities may dominate the linear regression model, leading to misleading coefficients that do not reflect actual causal relationships. After deseasonalizing, daily water levels looked plotted as shown in the picture on the left. Furthermore, statistical assessment (with ADF and KPSS testing) revealed that the water levels were stationary now, and therefore meet the assumption of ARMA. Hence, there



was no need to detrend the target variable further, as it would unnecessarily remove essential variability that contributes to the model's predictive power. However, linear regression models assume a linear relationship between features and the target variable. Unrelated trends in the features can dominate the model, resulting in misleading coefficients that do not reflect the true relationships. As it turned out by looking at the ADF test, climate change was the only feature that exhibited a strong trend. A correlation coefficient of 0.11 indicated a weak linear relationship between the long-term trends of water levels and climate change, the trend in climate change does not meaningfully explain the variability in water levels. To prevent the model from misinterpreting this unrelated trend as a significant predictor, we detrended the climate change feature, ensuring the focus remained on meaningful relationships with the target variable.

3. Modelling & Evaluation

a. ARMA

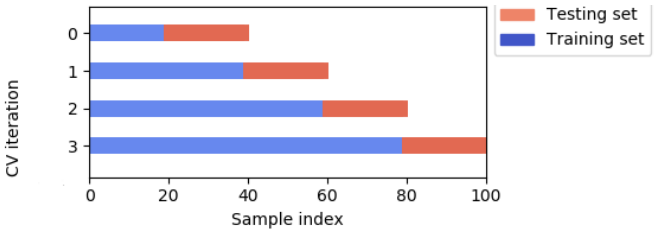
As ARMA models are statistical models designed to analyze and forecast only the target based on its past values and moving averages of residuals, all features were dropped. To identify the optimal hyperparameters, a grid search was performed on the training set over various combinations of the p (Autoregressive Order) and q (Moving Average Order) parameters. This process aimed to minimize the root mean squared error (RMSE) through Time Series Cross-Validation, ensuring the model achieves high predictive accuracy while avoiding overfitting. This procedure found the optimal hyperparameters to be p=4 and q=3. This means that the model uses the last 4 lagged observations of the water levels to predict the current value and the last 3 lagged residuals to adjust its predictions. After fitting the model to the training data, rolling forecasting (forecast for the next day only and iteratively) was performed on the test data using the fitted model. This forecasting was chosen as we cannot predict multiple steps ahead with ARMA without needing to update the input iteratively, since ARMA models rely on lagged observations and past forecast errors to make predictions. Hence, for accurate forecasting, the model needs the actual observed values (or its own past predictions) at each step.

b. Linear regression (Ridge + Lasso)

To enhance predictive modeling, Ridge and Lasso regression models were implemented. Both approaches address multicollinearity and regularize coefficients to prevent overfitting.

First, we prepared the already deseasoned and detrended data. We applied a StandardScaler to both features and the target variable to standardize data. Thus, we used penalties proportional to the magnitude of coefficients.

Second, we proceeded with the model implementation and the hyperparameter tuning. Ridge and Lasso regression models were trained using a grid search method with cross-validation (*GridSearchCV*) over a TimeSeriesSplit configuration. This ensured realistic evaluation as it maintained the temporal structure of the data, while preventing future data leakage into training splits.



The **Ridge regression** model minimizes the residual sum of squares, and adds a penalty on the squared magnitude of coefficients. The optimal *alpha* was found to be *0.006*. The latter balanced the penalization of large coefficients and the maintenance of the model's predictive power. **Lasso regression** applies an absolute value penalty to coefficients. It potentially sets some to zero. This feature selection mechanism is useful when identifying the most impactful variables. The optimal *alpha* for Lasso was *0.0001*. This provided a good compromise between feature sparsity and performance.

c. Random Forest

The random forest was set up with Hyperparameter-Tuning using TimeSeriesSplits and *RandomizedSearchCV*, as in contrast to *GridSearchCV*, it randomly chooses parameter combinations from distributions and allows for faster computation of larger ranges of parameters. *GridSearchCV* produced worse results when the optimal combination wasn't in the hyperparameter space, and as we didn't know the best parameter range, the randomized search was the better option.

We chose *n_estimators = 50* while performing the hyperparameter tuning to decrease computation time and used *n_estimators = 500* to train the model with finalized parameters later, as a greater number of decision trees only positively impacts the performance.

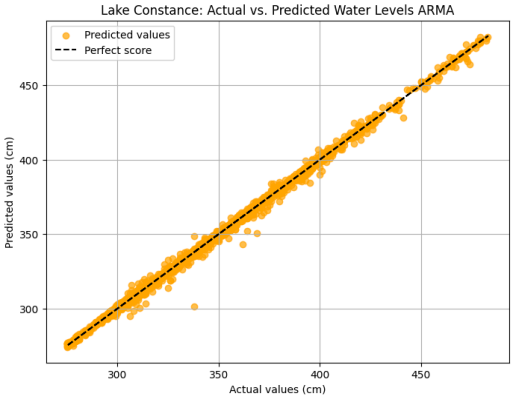
d. Neural network

For the neural network model we used the *MLPRegressor* by scikit-learn. Same as in the Random Forest model, hyperparameter tuning was done using *RandomizedSearchCV*, allowing for faster computation and also to explore a larger range of hyperparameters. As in random forest. This approach was particularly advantageous as the best parameter ranges were unknown at the start of the project. In the end, the best combination of parameters turned out to be 'adam' as solver, a learning rate of 0.0001, batch size 32, layer sizes (32, 64, 32) this parameter was particularly important as it allowed the model to capture intricate relationships in the data, while our alpha value of 0.01 helped to prevent overfitting. As the next step, we used these optimal values to train our final model.

Main Results

a. ARMA

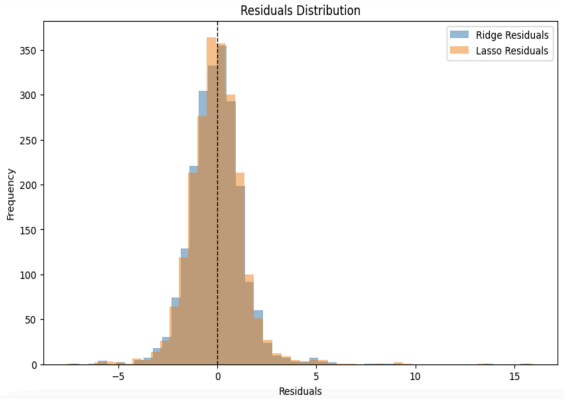
With an RMSE of 2.12, the ARMA model performed surprisingly well, given the simplicity of the model. Since ARMA assumes that the data can be explained using only its own lags and moving averages, it performs well when the target variable is driven primarily by internal dynamics rather than external factors (so past values heavily influence future value). This is supported by feature importance conducted for Neural Networks, where lagged values, particularly lag 1, have the highest influence, confirming the relevance of past observations in predicting water levels.



To evaluate the performance of the ARMA model, a baseline model was created using the previous day's water level. The RMSE for the ARMA model was 2.12, compared to 2.85 for the baseline, representing a ~25% improvement in prediction accuracy. In operational contexts like ferry management, this improvement translates to better decision-making, reduced costs, and enhanced safety and efficiency.

b. Linear Regression

The performance of Ridge and Lasso regression models was evaluated using metrics such as Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). **Ridge regression** showed an MSE of 1.96 and an RMSE of 1.40. These indicated a solid performance. **Lasso regression** recorded an MSE of about 2.04 and an RMSE of 1.43. These are slightly higher than Ridge but still competitive.



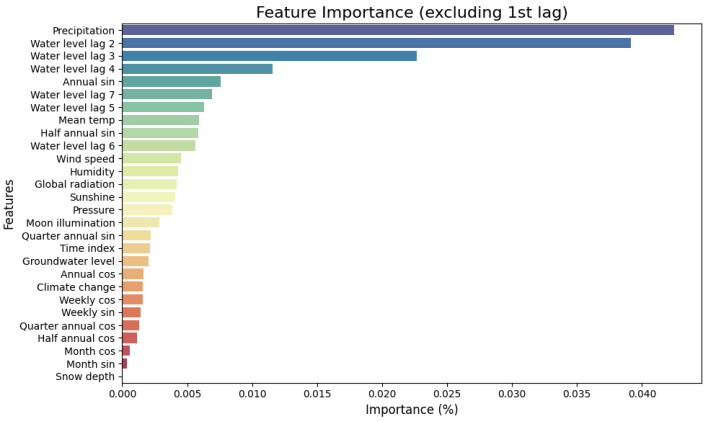
Visualization of the actual water levels against the predicted values revealed that both models closely followed the actual

trends. However, Ridge displayed a marginally better fit to the data. Additionally, a comparison of residual distributions (differences between actual and predicted values) showed that both models had residuals centered around zero. However, Ridge residuals are narrower, which indicates more accurate predictions.

In conclusion, both Ridge and Lasso regression models proved strong predictive capabilities. They offer reliable forecasting with interpretable coefficients and a low risk of overfitting. Nevertheless, Ridge regression slightly outperformed Lasso. Due to its ability to handle multicollinearity without aggressively "zeroing out" coefficients, its predictions are more consistent.

c. Random Forest

With an RMSE of 2.04, the random forest generally performed very well. Plotting the actual vs. predicted values, it could be seen that this was especially true for values between 320 to 360 cm.

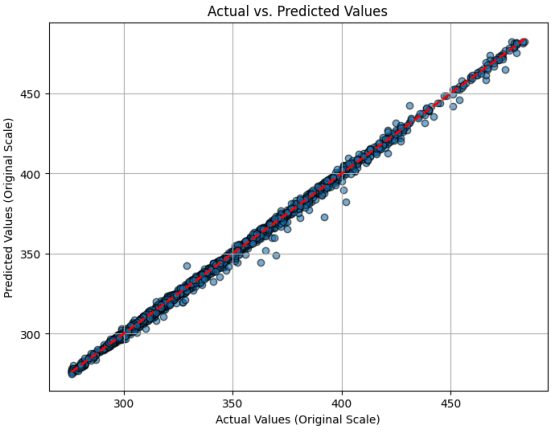


Being able to evaluate the importance of features is a great benefit of random forests as it allows for more transparency and comprehensibility. When looking at the feature importances, by far the most significant is the 1st lag (the water level of the previous day) with more than 99% importance.

In context this is very logical, as f.e. precipitation might be an indicator that the water level increased, but is useless for predicting the water level without knowing the previous value. To investigate the other features graphically, we then excluded the 1st lag.

It became clear that the precipitation and previous water levels are most important, as they feature the highest significance for the model. Snow depth on the other hand was not significant at all, probably because the model wasn't able to capture differences between snow depth levels (snow melting) and snowfall is already present in the precipitation.

d. Neural Network



For the Neural Network we evaluated the model using the RMSE, and found that it performed very well with a RMSE of 1.96. Even though neural networks do not inherently provide feature importance scores like Random Forest, we still attempted to do a feature importance analysis by looking at the permutation importance. Feature importance can be derived this way since permutation importance measures the impact of each feature on model performance by shuffling its values and observing the resulting drop in accuracy (increase in MSE). Here we found similar results as in the feature importance analysis for the other models, namely that the first water level lag was clearly the most crucial. While the precipitation feature also contributed significantly, for neural networks the other water level lags had even greater importance.

Results table

Model	ARMA	Linear Regression	Random Forest	Neural Network
RMSE	2.12	1.40 (Ridge) 1.43 (Lasso)	2.04	1.96