Data-driven models predict microbial water quality in the drinking water source using *E. coli* monitoring and hydrometeorological data

Ekaterina Sokolova\*, Mia Bondelind, Oscar Ivarsson, Ann Lillieström, Nora Speicher, Henrik Rydberg

\* corresponding author

# Introduction

Important to know microbial water quality to dimension drinking water treatment, including rapid changes in water quality. Laboratory analyses are available only with a delay in time. Therefore, it is of value to predict the microbial concentrations.

It has been shown that microbial water quality, e.g. E. coli, can be predicted using data-based models, e.g. X, Y and Z (Tornevi, Mohammed, …).

An earlier study has shown the relationship between the microbial concentrations in the Göta River and precipitation (Tornevi). In the case of the Göta River, apart the hydrometeorological data, at-line E. coli concentrations measured twice a day are available. However, these at-line measurements only provide categorical results, i.e. X, Y, Z.

Here, we explore whether at-line monitoring can be combined with hydrometeorological data in a data-driven model to predict concentrations of E. coli at the drinking water intake. We evaluate the ability of the models of different complexity to predict the concentrations at the water intake. Given the overall challenge of predicting microbial concentrations (References), we cannot expect the model to reproduce all peak values, however, from the operational standpoint, discrepancies between the predicted and measured values provide valuable information, as these may indicate events that need intervention, e.g. failures in the wastewater system. Th

The aim of this paper is to evaluate the suitability of data-driven models of varying complexity for predicting the concentrations of E. coli at the drinking water intake. The objectives are to (i) assess how the complexity of the model affects the model performance; and (ii) assess the predictive power of the explanatory variables.

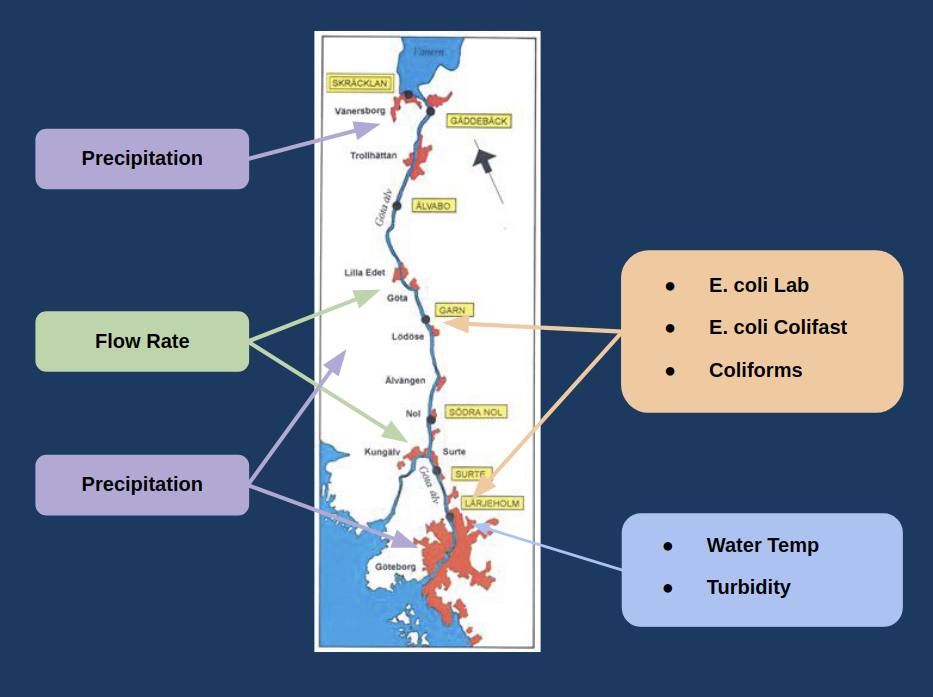
# Methodology

## Study area and data

The river Göta älv is located... and used for drinking water supply of … consumers.

The dataset used for model development is summarised in Table 1.

**Figure** showing the map with location of the measuring stations

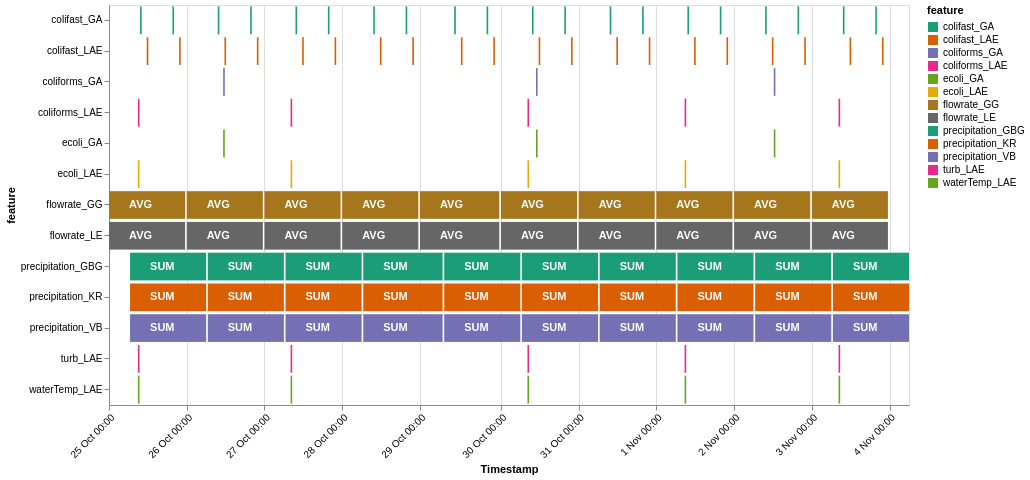


**Table 1** Input dataset

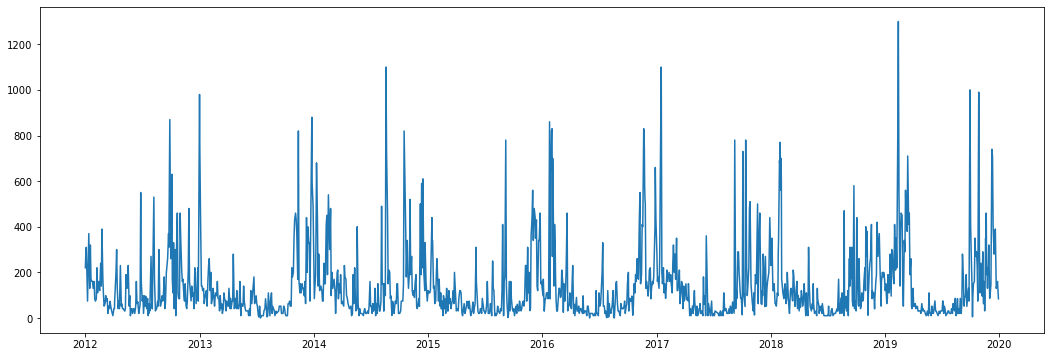
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Type | Location | Time period | Time resolution | Unit | Median | 5th percentile | 95th percentile | % Days Missing |
| E. coli Lab | Lärjeholm | 2012-01-02 – 2019-12-30 | Avg 2 days 6 hours | /100ml | 86.0 | 10.0 | 483.0 | 55.8 |
| E. coli Lab | Garn | 2012-01-02 – 2019-12-30 | Avg 3 days 3.5 hours | /100ml | 90.0 | 10.0 | 566.5 | 68.3 |
| E. coli Colifast | Lärjeholm | 2012-01-01 – 2019-12-31 | 2 times/day | /100ml |  |  |  | 4.0 |
| E. coli Colifast | Garn | 2012-02-16 – 2019-12-31 | 2 times/day | /100ml |  |  |  | 8.9 |
| Coliforms | Lärjeholm | 2012-01-02 – 2019-12-30 | Avg 2 days 4.5 hours | /100ml | 445.0 | 120.0 | 3325.0 | 54.4 |
| Coliforms | Garn | 2012-01-02 – 2019-12-30 | Avg 3 days 2.5 hours | /100ml | 470.0 | 110.0 | 3900.0 | 67.9 |
| Precipitation | Göteborg | 2002-01-01 – 2019-11-01 | Hourly Sum | mm | 0.0 | 0.0 | 0.6 | 2.2 |
| Precipitation | Vänersborg | 2002-01-01 – 2019-12-01 | Daily Sum | mm | 0.0 | 0.0 | 11.8 | 0.0 |
| Precipitation | Komperöd | 2002-01-01 – 2019-12-01 | Daily Sum | mm | 0.0 | 0.0 | 15.1 | 0.0 |
| Flow Rate | Lilla Edet | 2002-01-01 – 2020-01-29 | Daily Mean | m³/s | 550.0 | 206.0 | 900.0 | 0.0 |
| Flow Rate | Göteborgsgrenen | 2002-01-01 – 2020-01-29 | Daily Mean | m³/s | 152.0 | 132.0 | 225.0 | 0.0 |
| Water Temperature | Lärjeholm | 2012-01-02 – 2019-12-30 | Avg 2 days 1 hour | °C | 8.8 | 0.8 | 18.7 | 52.0 |
| Turbidity | Lärjeholm | 2012-01-02 – 2019-12-30 | Avg 2 days 7.5 hours | FNU | 6.25 | 3.3 | 16.4 | 57.3 |

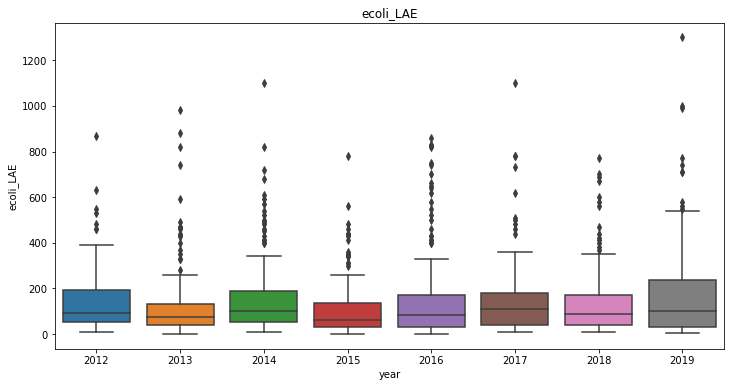
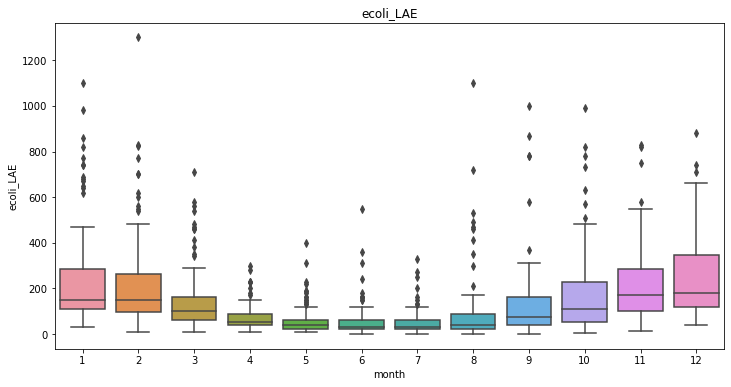
\*many values were missing and were filled by X

**Figure** plot showing the timing of the prediction and the timing of the input data



For combining different features to a dataset used as input for machine learning models we decide on using data from 2012-04-03 - 2019-10-30. Because we want to perform forecasting and the chronological order of the data, it is splitted into a training and test set with the test set being the last 2 years. For the target variable E. coli Lab (Lärjeholm) this gives us 872 observations in the training set and 341 observations in the test set.





By plotting the target variable, we can see that no obvious trend is present. The time series has a slight seasonal behaviour over a year (see plot X) with extreme values and large variation during the winter months. We expect that the main part of the seasonal behaviour can be explained by external predictors, e.g. water temperature.

The dataset consists of both regular and irregular features. The features for precipitation and flow rate are provided on a daily or hourly basis and thus regular. Other features consist of data sampled at irregular intervals where the time between observations could be up to several days. Dealing with combining regular and irregular time series can be done in various ways. In this project we have taken the approach to sample the irregular time series at a daily basis, leave days with no observations empty and use lags of observations as input to the models. The exception is the VAR model that requires the input format to be regularly spaced. In that case, forward filling is performed for the irregular features.

Basic data imputation is performed on the Colifast measurements since they have some days with missing data for the time period used in analysis. For the Colifast measurements, if data is missing for one day and measurements exist from the other location, then these values are used for imputing data. For the remaining missing values, forward fill is performed. The Colifast measurements are also provided as ordinal categorial values (<50, 50, 100, 200, 400, >400). To use these features in the analysis, they are simply transformed to numerical values that keeps the same order.

## Model set-up

The focus in this project has been on forecasting E. coli levels one day ahead using different type of lagged features. The Naive baseline, Exponential Smoothing and ARIMA model only makes use of E. coli data from Lärjeholm while the other methods include multiple other features as predictors. We use two different approaches, in the first we use default values for numerical microbial features (E. coli Lab and Coliforms) and in the second we perform Log-scaling using log(1+x). It should however be noted that in the hyperparameter optimization for default values we also try to transform the target variable (E. coli Lab) using the same type of log transformation and some results will use that transformation for the final model. The difference between the approaches in such case is that when log transformation is applied only on the target variable, the predictions are transformed back to their original scale so that the evaluation is performed on the original scale.

We have evaluated a number of different models of various complexity for the task of forecasting E. coli levels. The selection is based on that we wanted to investigate if better performance was achieved by using multivariate approaches as well as including more complex models that could capture non-linear relationships in the data.

The baseline approaches consist of univariate approaches that is common for modelling and forecasting time series data. We also include a few methods that is more common in the Machine Learning domain and could be used for various types of data and problems. To make these approaches work for time series problem we perform feature engineering with lagged features as input. Features with daily values also have daily lags and other features with non-daily measurements are defined with observation lags which mean that a lag of one will be the last observed value in the time series. We also add a multivariate autoregressive model in VAR (Vector Autoregression) to complement the ML approaches and see if it can achieve comparable performance.

Using the ML methods, the max lag that is used for each feature is selected based on domain knowledge and previous publications (Tornevi, 2014). The max lag for each feature is specified as following:

E. coli Lab - Lärjeholm: 3 Observations

E. coli Lab - Garn: 3 Observations

E. coli Colifast - Lärjeholm: 10 Observations

E. coli Colifast - Garn: 10 Observations

Coliforms - Lärjeholm: 3 Observations

Coliforms - Garn: 3 Observations

Water Temp - Lärjeholm: 1 Observation

Turbidity - Lärjeholm: 3 Observations

Precipitation - Göteborg: 6 Days

Precipitation - Komperöd: 6 Days

Precipitation - Vänersborg: 6 Days

Flow Rate - Göteborgsgrenen: 5 Days

Flow Rate - Lilla Edet: 5 Days  
  
**The Naive Model**   
  
The naive model uses the most recent observed value as its next prediction. This simple approach is used as a baseline for comparing the performance of more complex models.  
  
**Exponential Smoothing**  
  
Exponential Smoothing is a commonly used technique for the analysis of time-series data. It looks at past observations and weights their importance over time in an exponentially decreasing manner, such that the most recent observation has the highest weight. We use a variant called Simple Exponential Smoothing which is mostly suitable for data without trend or seasonality. In our case we have seasonality in the data and therefore an extension like Holt-Winters Exponential Smoothing could possibly be a way of improving this model. The method has a single parameter alpha that decides the smoothing factor, i.e. how much importance the model should allocate to its most recent value. The method implemented in the Python package statsmodel (<https://www.statsmodels.org/>) is used and the alpha parameter is optimized through maximizing the log-likelihood.  
  
**ARIMA - Auto-Regressive Integrated Moving Average**  
  
ARIMA is a kind of linear regression model that is auto-regressive, meaning that it uses previous values of the target variable to make its predictions. "Integrated" denotes the use of differencing to make the mean and variance consistent over time, resulting in a more robust model. "Moving average" means that the model forecasts a value using the model's past errors, i.e. it tries to correct the deviations between the past predicted and true values.

We have selected to include ARIMA because it is one of the most common methods for forecasting time series. The parameters (p,d,q) are used for selecting the order of the AR, I and MA terms in the model. The p and q parameters are optimized with the Akaike Information Criterion (AIC) and the parameter d is selected based on the KPSS-test for stationarity. For doing this, the Python package pyramid (<https://alkaline-ml.com/pmdarima>) is used. We have not tested the seasonal version of ARIMA because of computational reasons but that would be one potential extension for improving this model.  
  
**VAR - Vector Auto Regression**  
  
Vector auto regression is a generalization of the univariate autoregressive models, that allows for multivariate time-series. Each variable is modelled through a linear equation including the lagged values, the lagged values of the other variables and an error term. The lag parameter p is selected using AIC. The input data needs to formatted on a regular spaced format so in order to do this we simply use forward fill imputation of the irregular features to end up with daily values for all features. The VAR implementation in the Python package statsmodel (<https://www.statsmodels.org/>) is used.  
  
**LASSO (Least Absolute Shrinkage and Selection Operator)**  
  
LASSO is a regularization technique for regressive models. A regressive model aims to build a mathematical equation that defines the target variable as a function of the predictor variables. In general, regularization is implemented by adding a penalty to the best fit of the training data, in order to make the model generalize better on unseen test data. Regularization also restricts the influence of predictor variables by disallowing their coefficients to grow too large. LASSO uses the L1 regularization technique, which allows coefficients to become zero. In this way, LASSO can be used for automatically selecting relevant features. LASSO is selected because it is a method for identifying linear relationships in the data but can also cope with the problem of overfitting through regularization. It is also robust against multicollinearity in the data and analysis of the selected best subset of features can be performed.

LASSO har a parameter alpha that relates to how much regularization that should be applied. An alpha value of 0 therefore is equal to performing ordinary Linear Regression. This parameter is optimized using an expanding window version of the 5-fold cross validation. We also optimize with or without Log-transformation on the target variable and power scaling or standardization on the predictors. For LASSO we use the scikit-learn (<https://scikit-learn.org/>) package in Python.  
  
**Random forest**  
  
Random forests are a popular ensemble learning method, which work by combining multiple decision trees, and outputting their average prediction. A decision tree model builds a tree like structure from the observed data, where observations on the features of the data are represented by the branches of the tree and the conclusions are represented in the leaves. Random forests counteract overfitting of the data and error due to bias compared to individual decision trees. Random forest is good first choice to see if better performance can be achieved with a more complex model because it has been showed to work for a great number of tasks on various datasets. This method also has the ability of producing a feature importance score to see how much each feature contributes to the predicted values.

Here we use a similar cross validation setting as with LASSO and tune hyperparameters specific to Random Forest: Max depth, min sample per split, min sample per leaf, max ratio of features and if bootstrapping should be applied or not. We also try Log-transformation on the target variable but since these types of models are not influenced by transformations of the predictors, none is applied. For Random Forest we use the scikit-learn (<https://scikit-learn.org/>) package in Python.  
  
**TPOT (Tree-based Pipeline Optimization Tool)**  
  
TPOT is an automated machine learning library, which uses genetic programming to optimize machine learning pipelines. TPOT automates feature selection, model selection and parameter optimization, in order to find the best predictive model of the data at hand. The resulting model often tends to become quite complex and hard to interpret but it offers a way of estimating how much performance that is achievable through the use of much more complex architectures in an automatic setting. TPOT is provided through a Python package (<https://epistasislab.github.io/tpot/>).

## Model parameters

|  |  |  |
| --- | --- | --- |
|  | Default Values | Log-Scaled E. coli/Coliforms |
| ARIMA | p: 4  d: 0  q: 2 | p: 3  d: 0  q: 1 |
| VAR | Lag Order: 4 | Lag Order: 4 |
| LASSO | y Transformation: log(1+x)  y inverse Transformation: exp(x)-1  X Transformation: Power Transformation (Yeo-Johnson)  alpha: 0.026 | y Transformation: None  y inverse Transformation: None  X Transformation: Standard Scaler  alpha=0.025 |
| Random Forest | y Transformation: log(1+x)  y inverse Transformation: exp(x)-1  Bootstrap: False  Max Depth: 6  Max Features: 0.5  Min Samples per Leaf: 2  Min Samples per Split: 4  Nr of Estimators: 200 | y Transformation: None  y inverse Transformation: None  Bootstrap: True  Max Depth: 6  Max Features: 0.75  Min Samples per Leaf: 2  Min Samples per Split: 8  Nr of Estimators: 200 |
| TPOT | 1. MinMaxScaler() 2. StackingEstimator(estimator=ElasticNetCV(l1\_ratio=0.25, tol=0.1)) 3. StackingEstimator(estimator=XGBRegressor(learning\_rate=0.001, max\_depth=6, min\_child\_weight=10, n\_estimators=100, nthread=1, objective="reg:squarederror", subsample=0.35000000000000003)) 4. PolynomialFeatures(degree=2, include\_bias=False, interaction\_only=False) 5. StackingEstimator(estimator=SGDRegressor(alpha=0.0, eta0=0.01, fit\_intercept=False, l1\_ratio=0.5, learning\_rate="invscaling", loss="epsilon\_insensitive", penalty="elasticnet", power\_t=10.0)) 6. SelectFwe(score\_func=f\_regression, alpha=0.026000000000000002) 7. RandomForestRegressor(bootstrap=False, max\_features=0.35000000000000003, min\_samples\_leaf=18, min\_samples\_split=18, n\_estimators=100) | 1. StandardScaler() 2. MaxAbsScaler() 3. PCA(iterated\_power=1, svd\_solver="randomized") 4. StackingEstimator(estimator=ElasticNetCV(l1\_ratio=0.75, tol=0.1)) 5. PCA(iterated\_power=2, svd\_solver="randomized") 6. VarianceThreshold(threshold=0.005) 7. StackingEstimator(estimator=ElasticNetCV(l1\_ratio=0.35000000000000003, tol=0.1)) 8. SelectPercentile(score\_func=f\_regression, percentile=69) 9. StackingEstimator(estimator=XGBRegressor(learning\_rate=0.001, max\_depth=1, min\_child\_weight=8, n\_estimators=100, nthread=1, objective="reg:squarederror", subsample=0.2)) 10. StackingEstimator(estimator=SGDRegressor(alpha=0.01, eta0=0.1, fit\_intercept=False, l1\_ratio=0.25, learning\_rate="invscaling", loss="squared\_loss", penalty="elasticnet", power\_t=10.0)) 11. ElasticNetCV(l1\_ratio=0.8, tol=0.1) |

## Model performance

The performance of the different models was measured with a few common error metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Symmetrical Mean Absolute Percentage Error (SMAPE), an accuracy measure based on percentage errors. The symmetrical version of MAPE allows for observation value which are zero. Also, the coefficient of determination (R²) is used. The performance is measured on the train and validation splits during the cross validation as well as the unseen test data. Together with these metrics we also evaluate model performance by plotting forecasted predictions along with the true observations.

# Results

**Table X** Model performance on default values.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Train MAE | Test MAE | Train SMAPE | Test SMAPE | Train RMSE | Test RMSE | Train R² | Test R² |
| Naive | 89.84 | 93.07 | 66.19 | 64.16 | 159.71 | 159.06 | 0.11 | 0.18 |
| ExpSmooth | 84.63 | 85.41 | 62.00 | 58.61 | 143.28 | 144.17 | 0.28 | 0.33 |
| ARIMA | 83.81 | 89.08 | 65.50 | 66.37 | 136.50 | 141.68 | 0.35 | 0.35 |
| VAR | 65.73 | 71.59 | 59.19 | 63.56 | 111.86 | 116.90 | 0.55 | 0.56 |
| LASSO | 60.54 | 68.55 | 48.93 | 51.02 | 110.02 | 123.34 | 0.57 | 0.51 |
| RF | 36.40 | 71.50 | 34.21 | 51.99 | 67.02 | 129.59 | 0.84 | 0.46 |
| TPOT | 34.74 | 65.04 | 31.12 | 49.23 | 63.45 | 110.50 | 0.86 | 0.60 |

**Table X** Model performance on Log-scaled Microbial values.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Train MAE | Test MAE | Train SMAPE | Test SMAPE | Train RMSE | Test RMSE | Train R² | Test R² |
| Naive | 0.76 | 0.73 | 19.76 | 17.43 | 1.03 | 0.98 | 0.15 | 0.20 |
| ExpSmooth | 0.68 | 0.64 | 16.89 | 14.93 | 0.89 | 0.82 | 0.38 | 0.44 |
| ARIMA | 0.65 | 0.64 | 16.28 | 15.05 | 0.87 | 0.81 | 0.41 | 0.45 |
| VAR | 0.56 | 0.56 | 14.29 | 13.51 | 0.75 | 0.75 | 0.54 | 0.53 |
| LASSO | 0.50 | 0.51 | 12.91 | 12.31 | 0.67 | 0.66 | 0.63 | 0.64 |
| RF | 0.39 | 0.54 | 10.45 | 12.97 | 0.53 | 0.70 | 0.78 | 0.59 |
| TPOT | 0.51 | 0.53 | 12.99 | 12.70 | 0.68 | 0.69 | 0.62 | 0.61 |

**Figure 1** This figure would show for example one year during the testing period showing time-series of E. coli predicted by all (or several) models

**Feature Analysis**

A few of the models can be used for interpretation of the contribution of different features. For example, LASSO produces coefficients of the standardized variables that denotes the effect each variable has on the target variable. From the Random Forest method one can extract feature importance which is based on how much each feature contributes to decrease the impurity in the trees. This can later be used to rank the different features. One note to make is that these are calculated on the training data so with a model that is overfitting, the same features might not be contributing in the same way on unseen data.

**Table 4** Feature coefficients for the LASSO model on default E. coli/Coliform values.

|  |  |
| --- | --- |
| feature\_names | feature\_coef |
| ecoli\_colifast\_GA\_lag1O | 0.121459854844918 |
| precipitation\_VB\_lag2D | 0.11987302672143 |
| precipitation\_KR\_lag1D | 0.118739824002103 |
| ecoli\_colifast\_GA\_lag3O | 0.081848000641755 |
| coliforms\_GA\_lag1O | 0.081477613257634 |
| ecoli\_colifast\_LAE\_lag2O | 0.081393831831769 |
| ecoli\_colifast\_LAE\_lag1O | 0.080431832193037 |
| ecoli\_lab\_LAE\_lag2O | 0.080423666987765 |
| coliforms\_LAE\_lag1O | 0.064881731877485 |
| ecoli\_lab\_GA\_lag1O | 0.050084499420159 |
| ecoli\_colifast\_GA\_lag2O | 0.048257877406796 |
| ecoli\_lab\_GA\_lag2O | 0.048237632872142 |
| ecoli\_lab\_GA\_lag3O | 0.041294637778227 |
| ecoli\_colifast\_LAE\_lag9O | 0.037043459794923 |
| precipitation\_KR\_lag3D | 0.034173466276106 |
| ecoli\_lab\_LAE\_lag1O | 0.033389876066024 |
| precipitation\_KR\_lag2D | 0.030197542149098 |
| precipitation\_GBG\_lag2D | 0.029543518412772 |
| precipitation\_KR\_lag4D | 0.021869437983149 |
| ecoli\_colifast\_LAE\_lag8O | 0.012054952717233 |
| precipitation\_VB\_lag3D | 0.010121585786744 |
| ecoli\_colifast\_GA\_lag6O | 0.00532737395472 |
| ecoli\_colifast\_GA\_lag10O | 0.002743494476659 |
| ecoli\_colifast\_LAE\_lag4O | 0.00079585838228 |
| flowrate\_LE\_lag3D | 0 |
| coliforms\_GA\_lag2O | 0 |
| coliforms\_LAE\_lag3O | 0 |
| precipitation\_KR\_lag6D | 0 |
| flowrate\_LE\_lag1D | 0 |
| flowrate\_LE\_lag2D | 0 |
| coliforms\_LAE\_lag2O | 0 |
| turb\_LAE\_lag2O | 0 |
| turb\_LAE\_lag1O | 0 |
| flowrate\_LE\_lag5D | 0 |
| flowrate\_GG\_lag1D | 0 |
| turb\_LAE\_lag3O | 0 |
| flowrate\_GG\_lag2D | 0 |
| flowrate\_GG\_lag3D | 0 |
| flowrate\_GG\_lag4D | 0 |
| flowrate\_GG\_lag5D | 0 |
| precipitation\_KR\_lag5D | 0 |
| flowrate\_LE\_lag4D | 0 |
| precipitation\_VB\_lag1D | 0 |
| precipitation\_VB\_lag6D | 0 |
| ecoli\_colifast\_GA\_lag7O | 0 |
| ecoli\_lab\_LAE\_lag3O | 0 |
| ecoli\_colifast\_LAE\_lag3O | 0 |
| ecoli\_colifast\_LAE\_lag5O | 0 |
| ecoli\_colifast\_LAE\_lag6O | 0 |
| ecoli\_colifast\_LAE\_lag7O | 0 |
| ecoli\_colifast\_LAE\_lag10O | 0 |
| ecoli\_colifast\_GA\_lag4O | 0 |
| ecoli\_colifast\_GA\_lag5O | 0 |
| coliforms\_GA\_lag3O | 0 |
| ecoli\_colifast\_GA\_lag9O | 0 |
| precipitation\_GBG\_lag1D | 0 |
| precipitation\_GBG\_lag3D | 0 |
| precipitation\_GBG\_lag4D | 0 |
| precipitation\_GBG\_lag5D | 0 |
| precipitation\_GBG\_lag6D | 0 |
| precipitation\_VB\_lag4D | 0 |
| precipitation\_VB\_lag5D | 0 |
| ecoli\_colifast\_GA\_lag8O | -0.020132737312141 |
| waterTemp\_LAE\_lag1O | -0.314393686384384 |

**Table X** Feature coefficients for the LASSO model on Log scaled E. coli/Coliform values.

|  |  |
| --- | --- |
| feature\_names | feature\_coef |
| ecoli\_colifast\_GA\_lag1O | 0.119985264726873 |
| precipitation\_KR\_lag1D | 0.095644334291599 |
| precipitation\_KR\_lag2D | 0.090030663047486 |
| ecoli\_colifast\_GA\_lag3O | 0.09001637913051 |
| precipitation\_GBG\_lag2D | 0.084427073432296 |
| ecoli\_lab\_LAE\_lag2O | 0.082169000270046 |
| coliforms\_LAE\_lag1O | 0.076512348824207 |
| ecoli\_colifast\_LAE\_lag1O | 0.064918778939515 |
| coliforms\_GA\_lag1O | 0.062415128278397 |
| ecoli\_colifast\_LAE\_lag2O | 0.053578587003108 |
| precipitation\_VB\_lag3D | 0.050023609018049 |
| precipitation\_VB\_lag2D | 0.048219004804595 |
| ecoli\_lab\_LAE\_lag1O | 0.043041715754331 |
| ecoli\_lab\_GA\_lag3O | 0.042939266154991 |
| ecoli\_colifast\_LAE\_lag9O | 0.042346031924524 |
| precipitation\_GBG\_lag1D | 0.041467367345686 |
| ecoli\_colifast\_GA\_lag2O | 0.038658493408749 |
| ecoli\_lab\_GA\_lag2O | 0.032987261967042 |
| precipitation\_KR\_lag3D | 0.031190403520492 |
| ecoli\_lab\_GA\_lag1O | 0.030290872840232 |
| turb\_LAE\_lag3O | 0.013846912066329 |
| precipitation\_KR\_lag4D | 0.01277859765065 |
| ecoli\_colifast\_LAE\_lag4O | 0.008578977513217 |
| ecoli\_colifast\_GA\_lag6O | 0.007472519614855 |
| ecoli\_colifast\_LAE\_lag8O | 0.006402338958405 |
| ecoli\_colifast\_GA\_lag5O | 0.006329726882776 |
| ecoli\_colifast\_LAE\_lag3O | 0.003404489629305 |
| ecoli\_colifast\_GA\_lag4O | 0.002571668790737 |
| ecoli\_lab\_LAE\_lag3O | 0.000786547381467 |
| flowrate\_LE\_lag4D | 0 |
| flowrate\_LE\_lag5D | 0 |
| flowrate\_GG\_lag1D | 0 |
| flowrate\_LE\_lag3D | 0 |
| flowrate\_GG\_lag5D | 0 |
| flowrate\_GG\_lag2D | 0 |
| flowrate\_GG\_lag3D | 0 |
| flowrate\_GG\_lag4D | 0 |
| turb\_LAE\_lag1O | 0 |
| turb\_LAE\_lag2O | 0 |
| flowrate\_LE\_lag1D | 0 |
| coliforms\_GA\_lag2O | 0 |
| coliforms\_LAE\_lag3O | 0 |
| coliforms\_LAE\_lag2O | 0 |
| flowrate\_LE\_lag2D | 0 |
| precipitation\_VB\_lag1D | 0 |
| precipitation\_GBG\_lag3D | 0 |
| ecoli\_colifast\_GA\_lag10O | 0 |
| ecoli\_colifast\_LAE\_lag5O | 0 |
| ecoli\_colifast\_LAE\_lag6O | 0 |
| ecoli\_colifast\_LAE\_lag7O | 0 |
| ecoli\_colifast\_LAE\_lag10O | 0 |
| ecoli\_colifast\_GA\_lag7O | 0 |
| ecoli\_colifast\_GA\_lag8O | 0 |
| ecoli\_colifast\_GA\_lag9O | 0 |
| coliforms\_GA\_lag3O | 0 |
| precipitation\_KR\_lag6D | 0 |
| precipitation\_GBG\_lag4D | 0 |
| precipitation\_GBG\_lag5D | 0 |
| precipitation\_VB\_lag4D | 0 |
| precipitation\_VB\_lag5D | 0 |
| precipitation\_VB\_lag6D | 0 |
| precipitation\_KR\_lag5D | 0 |
| precipitation\_GBG\_lag6D | -0.000777939302133 |
| waterTemp\_LAE\_lag1O | -0.331775075929213 |

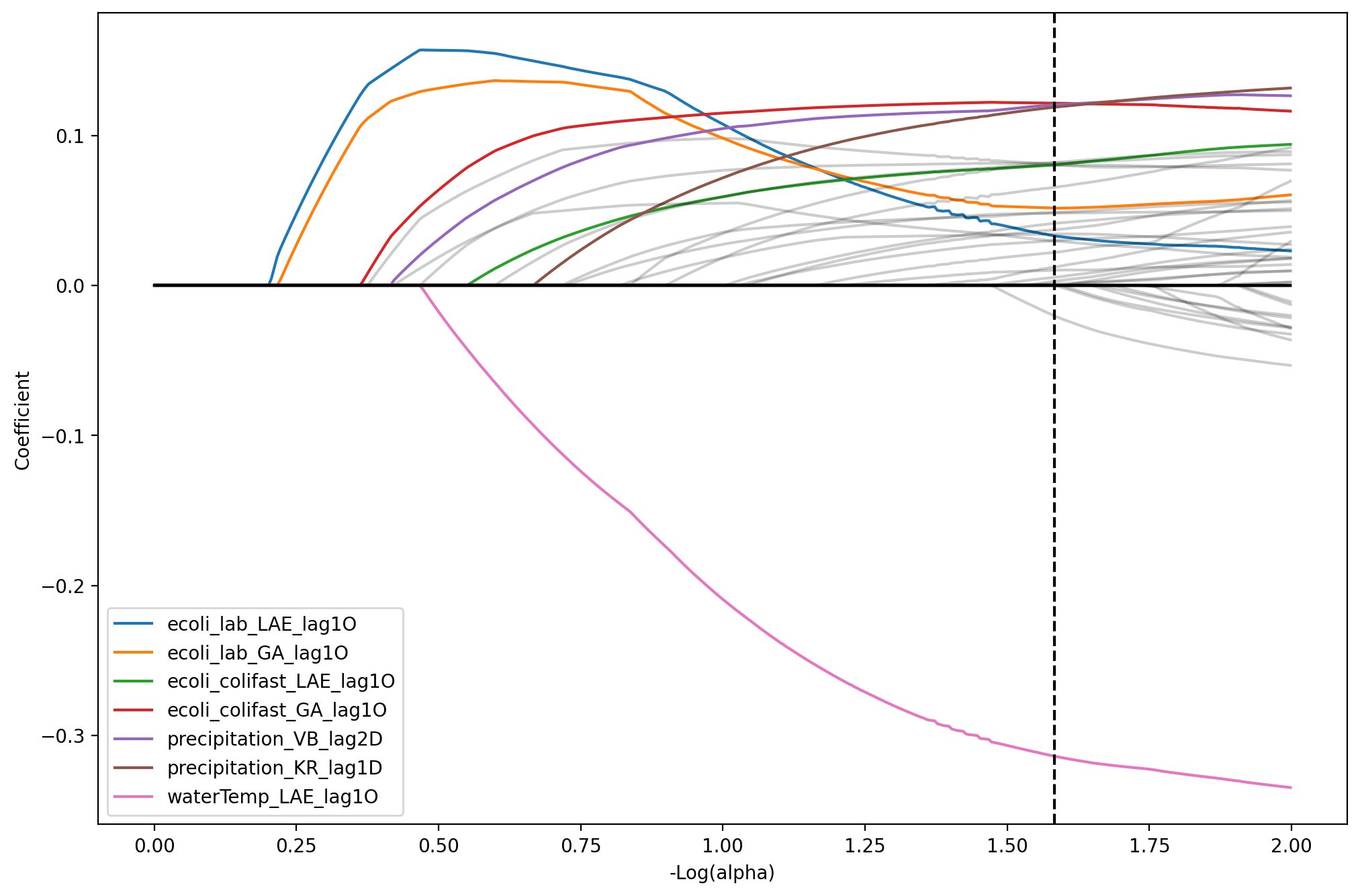
**Table X** Feature importance from the Random Forest model using default E. coli/Coliform values.

|  |  |
| --- | --- |
| feature\_names | feature\_importance |
| ecoli\_lab\_LAE\_lag1O | 0.212839864725604 |
| ecoli\_lab\_GA\_lag1O | 0.136736557991864 |
| waterTemp\_LAE\_lag1O | 0.08203788468798 |
| precipitation\_KR\_lag2D | 0.065676237837069 |
| ecoli\_colifast\_GA\_lag1O | 0.059367937525642 |
| ecoli\_lab\_LAE\_lag2O | 0.052542475000638 |
| precipitation\_VB\_lag2D | 0.039811212459793 |
| ecoli\_colifast\_GA\_lag3O | 0.032437350526018 |
| ecoli\_colifast\_LAE\_lag2O | 0.029616494935608 |
| coliforms\_GA\_lag1O | 0.025471371611314 |
| ecoli\_colifast\_LAE\_lag1O | 0.02138213368822 |
| precipitation\_KR\_lag1D | 0.021090760613109 |
| turb\_LAE\_lag2O | 0.01874348836277 |
| precipitation\_GBG\_lag2D | 0.014344974860771 |
| ecoli\_colifast\_GA\_lag2O | 0.009627826120309 |
| coliforms\_LAE\_lag1O | 0.009330044275795 |
| flowrate\_GG\_lag1D | 0.008907616265491 |
| ecoli\_colifast\_LAE\_lag3O | 0.008556093357948 |
| flowrate\_GG\_lag5D | 0.008352510763081 |
| precipitation\_GBG\_lag1D | 0.007983439195846 |
| turb\_LAE\_lag1O | 0.007633420464451 |
| turb\_LAE\_lag3O | 0.007523740411987 |
| precipitation\_VB\_lag1D | 0.00714723553975 |
| ecoli\_lab\_GA\_lag2O | 0.006761232811989 |
| flowrate\_GG\_lag4D | 0.00576469503752 |
| flowrate\_LE\_lag5D | 0.005612200373732 |
| precipitation\_VB\_lag3D | 0.005357045636615 |
| flowrate\_LE\_lag4D | 0.005217028555882 |
| flowrate\_LE\_lag1D | 0.005112508950603 |
| ecoli\_lab\_LAE\_lag3O | 0.004989413407126 |
| coliforms\_LAE\_lag3O | 0.004352902269665 |
| precipitation\_KR\_lag3D | 0.004199417088795 |
| flowrate\_LE\_lag2D | 0.003947389894308 |
| flowrate\_LE\_lag3D | 0.003841446742091 |
| ecoli\_lab\_GA\_lag3O | 0.003603986276551 |
| precipitation\_KR\_lag5D | 0.003475228367272 |
| flowrate\_GG\_lag2D | 0.003430444008647 |
| coliforms\_GA\_lag3O | 0.003280895868904 |
| coliforms\_GA\_lag2O | 0.003180203303177 |
| flowrate\_GG\_lag3D | 0.002930347674319 |
| coliforms\_LAE\_lag2O | 0.002800910429762 |
| precipitation\_GBG\_lag6D | 0.002760959128033 |
| ecoli\_colifast\_GA\_lag5O | 0.002481573209936 |
| precipitation\_GBG\_lag4D | 0.002207164142627 |
| precipitation\_GBG\_lag5D | 0.002079115232969 |
| precipitation\_VB\_lag4D | 0.00206169804656 |
| precipitation\_KR\_lag4D | 0.001894704763275 |
| ecoli\_colifast\_GA\_lag9O | 0.001785750137873 |
| ecoli\_colifast\_GA\_lag4O | 0.001694307971538 |
| ecoli\_colifast\_GA\_lag6O | 0.001659975527554 |
| precipitation\_GBG\_lag3D | 0.001610821633471 |
| precipitation\_VB\_lag5D | 0.001557526838905 |
| ecoli\_colifast\_LAE\_lag4O | 0.00150864373579 |
| ecoli\_colifast\_LAE\_lag9O | 0.001377136490861 |
| precipitation\_VB\_lag6D | 0.001374463463205 |
| ecoli\_colifast\_LAE\_lag7O | 0.001336308304276 |
| ecoli\_colifast\_GA\_lag7O | 0.001161708942378 |
| ecoli\_colifast\_GA\_lag10O | 0.001123112416425 |
| ecoli\_colifast\_LAE\_lag5O | 0.001027519632017 |
| ecoli\_colifast\_LAE\_lag8O | 0.000963545597786 |
| ecoli\_colifast\_GA\_lag8O | 0.000896100484012 |
| precipitation\_KR\_lag6D | 0.000875936388036 |
| ecoli\_colifast\_LAE\_lag6O | 0.000838080265664 |
| ecoli\_colifast\_LAE\_lag10O | 0.000705879728791 |

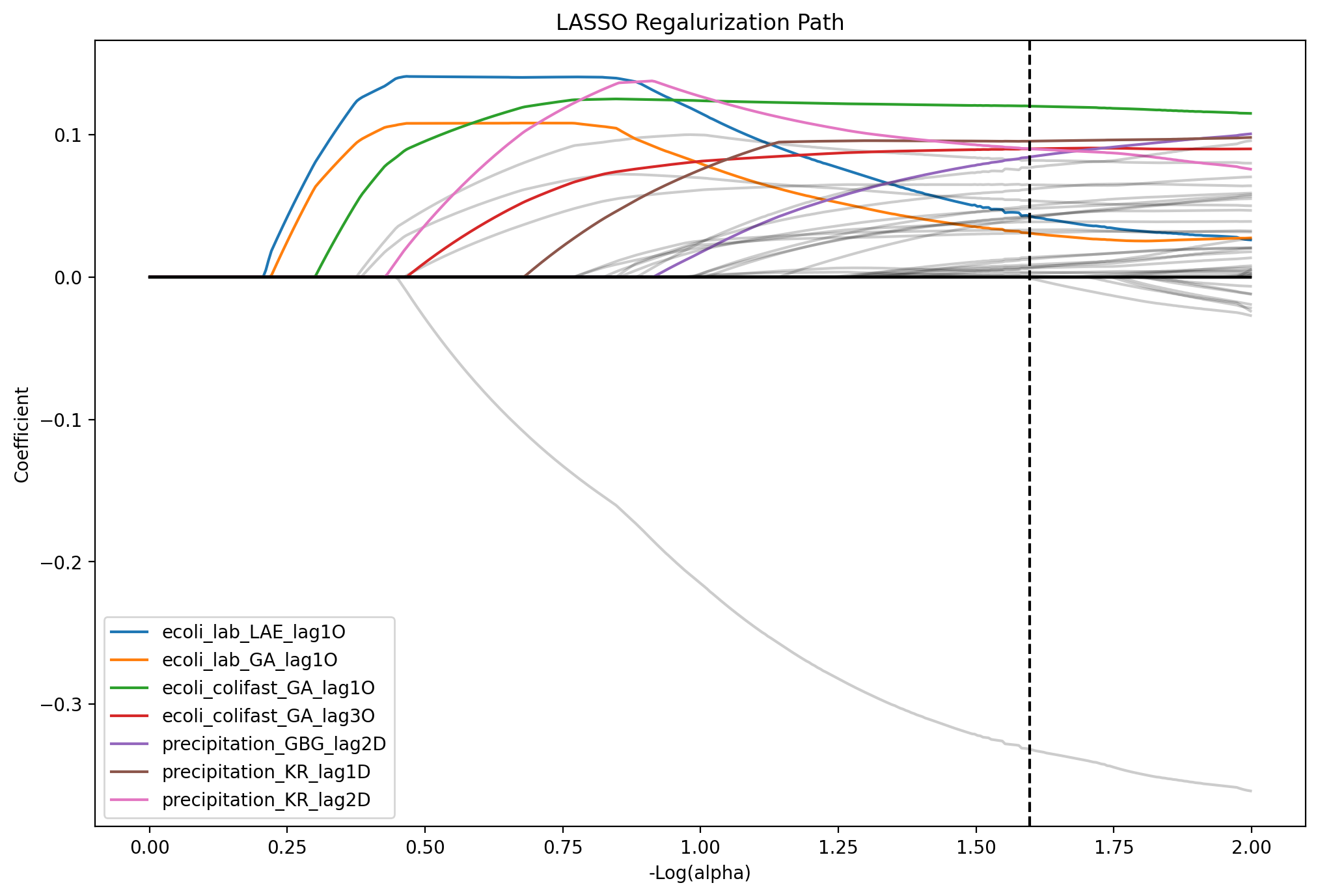
**Table X** Feature importance from the Random Forest model using Log scaled E. coli/Coliform values.

|  |  |
| --- | --- |
| feature\_names | feature\_importance |
| ecoli\_lab\_LAE\_lag1O | 0.236143853320049 |
| ecoli\_lab\_GA\_lag1O | 0.147881201938925 |
| waterTemp\_LAE\_lag1O | 0.086158199002146 |
| precipitation\_KR\_lag2D | 0.060009131518038 |
| ecoli\_colifast\_GA\_lag1O | 0.045139139648252 |
| ecoli\_lab\_LAE\_lag2O | 0.038082291764959 |
| precipitation\_VB\_lag2D | 0.03629388530955 |
| ecoli\_colifast\_GA\_lag3O | 0.025194785069308 |
| precipitation\_KR\_lag1D | 0.022091046306718 |
| coliforms\_GA\_lag1O | 0.02104895239902 |
| ecoli\_colifast\_LAE\_lag2O | 0.020968613931981 |
| ecoli\_colifast\_LAE\_lag1O | 0.017496628476335 |
| precipitation\_GBG\_lag2D | 0.016901410957483 |
| coliforms\_LAE\_lag1O | 0.015105351203834 |
| turb\_LAE\_lag2O | 0.013765189484067 |
| ecoli\_colifast\_GA\_lag2O | 0.010370498388134 |
| ecoli\_lab\_GA\_lag2O | 0.0099302699371 |
| turb\_LAE\_lag1O | 0.008983270811016 |
| turb\_LAE\_lag3O | 0.008795014123197 |
| coliforms\_LAE\_lag3O | 0.00863610356356 |
| precipitation\_GBG\_lag1D | 0.008113006539758 |
| flowrate\_GG\_lag1D | 0.006974014414746 |
| precipitation\_KR\_lag3D | 0.006704855311758 |
| precipitation\_VB\_lag1D | 0.006240751308994 |
| flowrate\_LE\_lag1D | 0.006090537630377 |
| flowrate\_GG\_lag4D | 0.006050704024557 |
| ecoli\_lab\_LAE\_lag3O | 0.005722585094022 |
| precipitation\_VB\_lag3D | 0.005607364432684 |
| coliforms\_GA\_lag3O | 0.005516226388544 |
| ecoli\_lab\_GA\_lag3O | 0.005293764090399 |
| flowrate\_LE\_lag2D | 0.005256534589444 |
| flowrate\_GG\_lag5D | 0.005139424116466 |
| coliforms\_LAE\_lag2O | 0.005028352048361 |
| ecoli\_colifast\_LAE\_lag3O | 0.004987786881433 |
| flowrate\_LE\_lag5D | 0.004695626310866 |
| coliforms\_GA\_lag2O | 0.00386806636985 |
| flowrate\_LE\_lag3D | 0.003803355992728 |
| flowrate\_LE\_lag4D | 0.003421694639381 |
| precipitation\_GBG\_lag5D | 0.003260940996187 |
| precipitation\_VB\_lag5D | 0.003138082754761 |
| precipitation\_VB\_lag4D | 0.002977988413796 |
| precipitation\_KR\_lag5D | 0.002926711652168 |
| flowrate\_GG\_lag3D | 0.002884626003435 |
| ecoli\_colifast\_GA\_lag4O | 0.00265510084884 |
| precipitation\_KR\_lag4D | 0.002561381993059 |
| flowrate\_GG\_lag2D | 0.002525571864114 |
| precipitation\_GBG\_lag6D | 0.002480410499352 |
| precipitation\_GBG\_lag3D | 0.002445966990325 |
| ecoli\_colifast\_GA\_lag5O | 0.002008797300107 |
| ecoli\_colifast\_GA\_lag10O | 0.001998210481733 |
| ecoli\_colifast\_LAE\_lag4O | 0.001951021845204 |
| ecoli\_colifast\_LAE\_lag5O | 0.001799469559589 |
| ecoli\_colifast\_LAE\_lag10O | 0.0017193410783 |
| ecoli\_colifast\_GA\_lag7O | 0.001717107621276 |
| precipitation\_VB\_lag6D | 0.001681841464171 |
| precipitation\_GBG\_lag4D | 0.001646636901479 |
| precipitation\_KR\_lag6D | 0.001612529786328 |
| ecoli\_colifast\_GA\_lag6O | 0.001584429229275 |
| ecoli\_colifast\_GA\_lag9O | 0.001559467544817 |
| ecoli\_colifast\_LAE\_lag7O | 0.001443058493001 |
| ecoli\_colifast\_LAE\_lag9O | 0.001348361003336 |
| ecoli\_colifast\_GA\_lag8O | 0.001113226159619 |
| ecoli\_colifast\_LAE\_lag8O | 0.00078366861199 |
| ecoli\_colifast\_LAE\_lag6O | 0.000666563565697 |

**Figure X** Lasso Regularization Path for default E. coli/Coliforms values. Best model is plotted with vertical dashed line. A few of the most relevant features are appear in colour.



**Figure X** Lasso Regularization Path for Log scaled E. coli/Coliforms values. Best model is plotted with vertical dashed line. A few of the most relevant features appear in colour.



The LASSO regularization path can be used to show what effect different values of the regularization parameter alpha has one the coefficients. One the far left on the x-axis, much regularization is applied and thus has the effect to put all the coefficients to zero. On the far right, little regularization is applied and the model introduces more and more features with non-zero coefficients.

False positive, false negatives.

* + Model performance
  + Performance parameters
  + Performance in Open/close intake

# Discussion

Discuss how the model results change with the increased complexity.

Discuss which features (predictive variables) explain the results.

Way to improve the model(s). Can the model be improved focussing on avoiding false negatives?

Discuss which other model or methods could be relevant to test in the future.

Discuss our results in relation to other publications (Tornevi, Mohammed, …)

Discuss the significance of our results for the drinking water supply.

Based on the performance of the different methods we can see they are quite consistent in their ranking among the different metrics used. Initially we can see that we improve on the naive baseline that we setup, indicating that there are historical patterns in the data that can be used for making a better forecasting for the next day. If we compare Exponential Smoothing and ARIMA, they perform quite similar with ExpSmooth having better performance on MAE and SMAPE while ARIMA performs slightly better on RMSE and R² on the default values. Since MAE and SMAPE has a greater effect on smaller errors and RMSE has a greater effect on large errors this could mean that ARIMA performs worse during periods with lower levels of E. coli and could potentially be improved by a seasonal component in the model.

By including external predictors, we can see the performance becomes better with the best performance for the default values achieved by the TPOT library. This method however produces very complex pipelines and by comparing the test scores with training scores, we can see that there is a gap indicating overfitting. That is also the case for the Random Forest method. Even though we constrain the complexity of the Random Forest method, e.g. by not making the trees too deep, the model is prone to overfit and thus perform much better on the training data than on the unseen test data. This also means that we should be a little bit careful when analysing the feature importances of that model since they don't exactly describe what features that are important in making predictions on unseen data.

Overall, it doesn't seem like using methods that includes non-linearities in their modelling (Random Forest, TPOT) achieves that much better than the linear multivariate methods (LASSO, VAR). Since these methods also are harder to interpret it would maybe be a bad choice for a method to use in practice. One reason why they don't perform better and are prone to overfit could simply be because they would need more data to perform well.

We can also see that by log scaling the microbial features LASSO performs best on all metrics for the test data. This method clearly benefits from reducing skewness in the distribution of these features.

As can be seen in the plot of the LASSO regularization path, the first two features introduced by the model (blue and orange line) with E. coli measurements from lab analysis are not the ones with the highest effect on the target variable in the best model (plotted with a vertical dashed line). Instead, when more features are introduced, they change their slope and eventually starts to degrade. The change in slope can probably be explained with multicollinearity between several of the features. When new features are introduced into the model and they are highly correlated to already existing features, that will affect the coefficients. One idea that the best model is achieved with Colifast measurements instead of the more accurate lab analysis of E. coli would be that the Colifast observations appear more frequently. The latest Colifast observation when making a forecast is often in the afternoon from the day before while an observation from a lab analysis could be several days past. We can also see from the coefficients that water temperature has a large negative effect, possibly explaining the seasonality in the data with lower levels of E. coli when the water temperatures are higher during the summer months.

Further work on extending the univariate methods would be to also include seasonal components to see if better performance could be achieved in this way. Also, additional imputation techniques could be investigated to deal with the problem of irregular time series for these methods. Another approach would also be to investigate methods that could deal with irregular time series internally. If prediction of peaks in E. coli levels during the winter months is the most important case, one can also develop a specialized model for that specific purpose and see if such a model could perform better. For example, one could make the problem into a classification task for such peaks. One thing that also could be investigated is models that also estimate uncertainty in their predictions. That would probably be of importance if a model is going to be used in practice. Then you would like to receive info about how sure the model is in its prediction.

# Conclusions

# References

Seabold, Skipper, and Josef Perktold. “[statsmodels: Econometric and statistical modeling with python.](http://conference.scipy.org/proceedings/scipy2010/pdfs/seabold.pdf)” *Proceedings of the 9th Python in Science Conference.* 2010.

[Scikit-learn: Machine Learning in Python](http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html), Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

Trang T. Le, Weixuan Fu and Jason H. Moore (2020). [Scaling tree-based automated machine learning to biomedical big data with a feature set selector](https://academic.oup.com/bioinformatics/article/36/1/250/5511404). *Bioinformatics*.36(1): 250-256.