Predict FloodHealthIndex (regression)

Performance modeling of FloodHealthIndex on DataWithGeo\_location\_prepared

short line



|  |  |  |
| --- | --- | --- |
| **Version** | **Author** | **Date** |
| 1.0 | bbo  no email | 2022-10-09 14:48:03 |

Table of contents

[I. Executive Summary 4](#_Toc44339452)

# Executive Summary

A Regression Machine Learning model was built using Dataiku DSS Visual ML. Its goal is to predict FloodHealthIndex given a total of 4 features. Using a dataset of 5924 rows, the process led to the selection of the LightGBM algorithm.

## Methodology

To ensure a good generalization capability for the ML model, a test strategy was set up. Data on which the ML model was not trained on was used for this purpose. The testing strategy was the following:

|  |  |
| --- | --- |
| Policy | Split the dataset |
| Use time ordering | No |
| Sampling method | First records |
| Record limit | 100000 |
| Split mode | Random |
| Use K-fold cross-testing | No |
| Train ratio | 0.8 |
| Random seed | 1337 |

See section [II.E](#_l7vf8w6y5ebt) for detailed explanations about these options.

Before being tested, the ML model has been tuned to find the best combination of hyperparameters according to the R2 Score metric. This optimal hyperparameter search, based on assessing performance on a validation set, was done using the following methodology:

|  |  |
| --- | --- |
| Search strategy |  |
| Strategy | Random search |
| Search parameters |  |
| Random state (hyperparameter search) | 1337 |
| Max number of iterations | 24 |
| Max search time | 0 (no limit) |
| Parallelism | 4 |
| Cross-validation |  |
| Cross-validation strategy | K-fold |
| Number of folds | 5 |
| Random state (cross-validation split) | 1337 |

See section II.D.3 for detailed explanations about these options.

## Results

The LightGBM algorithm was selected. The evaluation metric used to tune the hyperparameters was R2 Score computed on the validation dataset. After the best hyperparameter combination was found, the same metric was also computed on the test dataset. The final value was 0.931.

# Methodology

This section deals with the methodological details:

* The *Problem Definition* consists of selecting the target (**FloodHealthIndex**) and the type of problem (Regression).
* *Data Ingestion* analyzes each feature in order to maximize its prediction potential.
* *Model and Feature Tuning* finds the best hyperparameter set for the selected algorithm.
* The *Model Evaluation and Selection* strategy indicates how to compute the metrics that allows to evaluate the performance of the model.

## Problem Definition

A Regression Machine-Learning model was built using Dataiku DSS. Its goal is to predict **FloodHealthIndex** given a total of 4 features.

## Data Ingestion

During the data ingestion phase, the features are transformed into numerical features without missing values so as to be ingestible by the Machine Learning algorithm. The table below summarizes the processing applied to each of them.

|  |  |  |  |
| --- | --- | --- | --- |
| Feature Name | Status | Type | Processing |
| geo | Rejected | Category |  |
| FloodHealthIndex | Target | Numeric |  |
| INTPTLON10 | Input | Numeric | Avg-std rescaling |
| INTPTLAT10 | Input | Numeric | Avg-std rescaling |

|  |
| --- |
| **Legend**   * *Feature name:* Name of the feature column * *Feature status:* Input, Target or Rejected * *Feature type:* Numeric, Category, Text, or Array * *Processing:* Type of processing applied (Avg-std rescaling, dummy-encode…) |

## Model and Feature Tuning

### Pre-processings

Once each feature has been processed, it is possible to combine them to generate new features:

* Pairwise linear feature generation: Disabled
* Pairwise polynomial feature generation (A\*B) for all pairs of features: Disabled

### Tested Algorithm

The LightGBM algorithm has been tested.

|  |
| --- |
| LightGBM is a tree-based gradient boosting library designed to be distributed and efficient. This algorithm provides fast training speed, low memory usage, good accuracy and is capable of handling large scale data.  For more information on gradient tree boosting, see the "Gradient tree boosting" algorithm. |

The settings for this algorithm are given below. For hyperparameters, the possible values or ranges are listed:

|  |  |
| --- | --- |
| Boosting type | Try Gradient Boosting Decision Tree: Yes Try Gradient One-Side sampling: No |
| Maximum number of trees | Min: 50 Max: 200 Uniform distribution |
| Maximum depth of trees | -1 |
| Number of leaves | Min: 20 Max: 500 Uniform distribution |
| Learning rate | Min: 0.1 Max: 0.6 Uniform distribution |
| Minimum split gain | Min: 0 Max: 1 Uniform distribution |
| Minimum child weight | Min: 0.001 Max: 1 Uniform distribution |
| Minimum leaf samples | Min: 1 Max: 100 Uniform distribution |
| Columns subsample ratio for trees | Min: 0.5 Max: 1 Uniform distribution |
| L1 regularization | Min: 0 Max: 1 Uniform distribution |
| L2 regularization | Min: 0 Max: 1 Uniform distribution |
| Use bagging | Yes |
| Subsample ratio | 0.75 |
| Subsample frequency | 2 |
| Early stopping | Yes |
| Early stopping rounds | 4 |
| Random state | 1337 |
| Parallelism | 4 |

### Hyperparameter Search

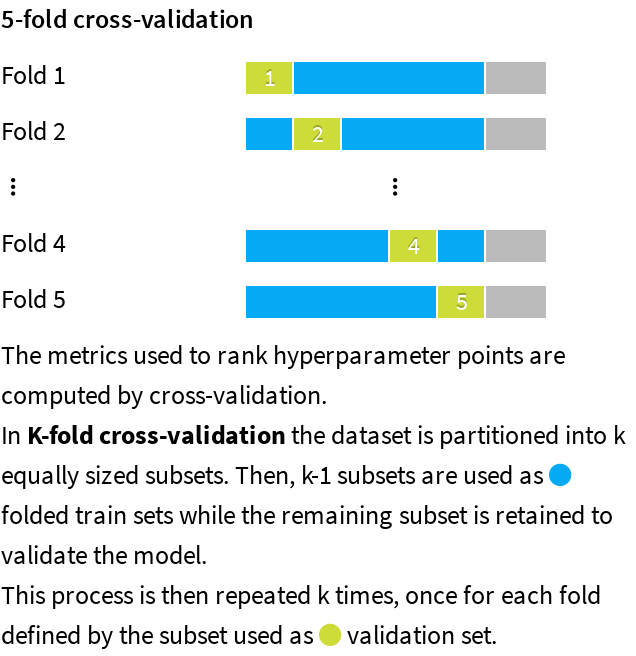
The hyperparameter search is done for each algorithm separately. It consists of finding the combination of hyperparameters that results in the best-trained model according to the validation metric (R2 Score) computed on the validation dataset.

The actual search settings for the selected algorithm are the following:

|  |  |
| --- | --- |
| Search strategy |  |
| Strategy | Random search |
| Search parameters |  |
| Random state (hyperparameter search) | 1337 |
| Max number of iterations | 24 |
| Max search time | 0 (no limit) |
| Parallelism | 4 |
| Cross-validation |  |
| Cross-validation strategy | K-fold |
| Number of folds | 5 |
| Random state (cross-validation split) | 1337 |

|  |
| --- |
| **Legend**   * *Randomize grid search:* If true, the grid was shuffled before the search. * *Max number of iterations:* This parameter sets the number of points of the grid that have been evaluated. * *Max search time:* Maximum search time in minutes. * *Parallelism:* -1 for automatic. It sets the number of hyperparameter searches that are performed simultaneously. * *Stratified:* If true, the same target distribution is kept in all the splits. |

Illustration:



*Note:* A grey area appears on the graphic to illustrate the data that is used for the test dataset.

## Evaluation and Selection

The last part of the methodology consists of comparing the performance of each algorithm trained using the best hyperparameter combination. The policy can consist in either:

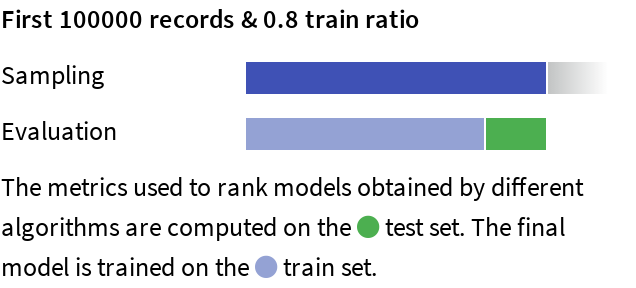
* Splitting the dataset by setting apart a test dataset, also called the hold-out dataset, for this performance evaluation. The train ratio indicates the amount of the dataset used in training, the remaining being used for evaluation.
* Performing a K-fold evaluation. It allows a more precise performance evaluation, at the expense of increased computation time.

This is indicated by the policy and the split mode in the table below.

When the original dataset is very big, the required computational resources may be too large compared to the expected benefit of training algorithms on it. As a result, the training, validation, and testing may be performed on a subset of the dataset. The sampling method given in the table below defines how it is built.

|  |  |
| --- | --- |
| Policy | Split the dataset |
| Use time ordering | No |
| Sampling method | First records |
| Record limit | 100000 |
| Split mode | Random |
| Use K-fold cross-testing | No |
| Train ratio | 0.8 |
| Random seed | 1337 |

Illustration:



|  |
| --- |
| **Legend**   * Policy:   + *Split the dataset:* Split a subset of the dataset.   + *Explicit extracts from the dataset:* Use two extracts from the dataset, one for the train set, one for the test set.   + *Explicit extracts from two datasets:* Use two extracts from two different datasets, one for the train set, one for the test set.   + *Split another dataset:* Split a subset of another dataset, compatible with the dataset.   + *Explicit extracts from another dataset:* Use two extracts from another dataset, one for the train set, one for the test set. * *Sampling method:* A subset may have been extracted in order to limit the computational resources required by the evaluation and selection process. The *Record limit* gives its size.   + *No sampling (whole data)*: the complete dataset has been kept.   + *First records*: The first N rows of the dataset have been kept (or all the dataset if it has fewer rows. The current dataset has 5924 rows). It may result in a very biased view of the dataset.   + *Random (approx. ratio)*: Randomly selects approximately X% of the rows.   + *Random (approx. nb. records)*: Randomly selects approximately N rows.   + *Column values subset (approx. nb. records)*: Randomly selects a subset of values and chooses all rows with these values, in order to obtain approximately N rows. This is useful for selecting a subset of customers, for example.   + *Class rebalance (approx. nb. records)*: Randomly selects approximately N rows, trying to rebalance equally all modalities of a column. It does not oversample, only undersamples (so some rare modalities may remain under-represented). Rebalancing is not exact.   + *Class rebalance (approx. ratio)*: Randomly selects approximately X% of the rows, trying to rebalance equally all modalities of a column. It does not oversample, only undersamples (so some rare modalities may remain under-represented). Rebalancing is not exact. * Partitions:   + *All partitions:* Use all partitions of the dataset.   + *Select partitions:* Use an explicitly selected list of partitions.   + *Latest partition:* Use the “latest” partition currently available in the dataset. “Latest” is only defined for single-dimension time-based partitioning. * *Time variable:* By enabling time-based ordering, DSS checks that the train and the test sets are consistent with the time variable. Moreover, DSS guarantees that:   + The train set is sorted according to the selected variable.   + The hyperparameter search is done with training sets and validation sets consistent with the ordering induced by the time variable. * *Split mode:* If “*K-fold cross-test*” is selected, it gives error margins on metrics, but strongly increases training time. * *Train ratio:* Proportion of the sample that goes to the train set. The rest goes to the test set. * *Number of folds:* Number of folds K to divide the dataset into. * *Random seed:* Using a fixed random seed allows for reproducible results. |

# Experiment Results

The methodology detailed in the previous section has been run. The obtained results are presented in this section.

## Selected Model

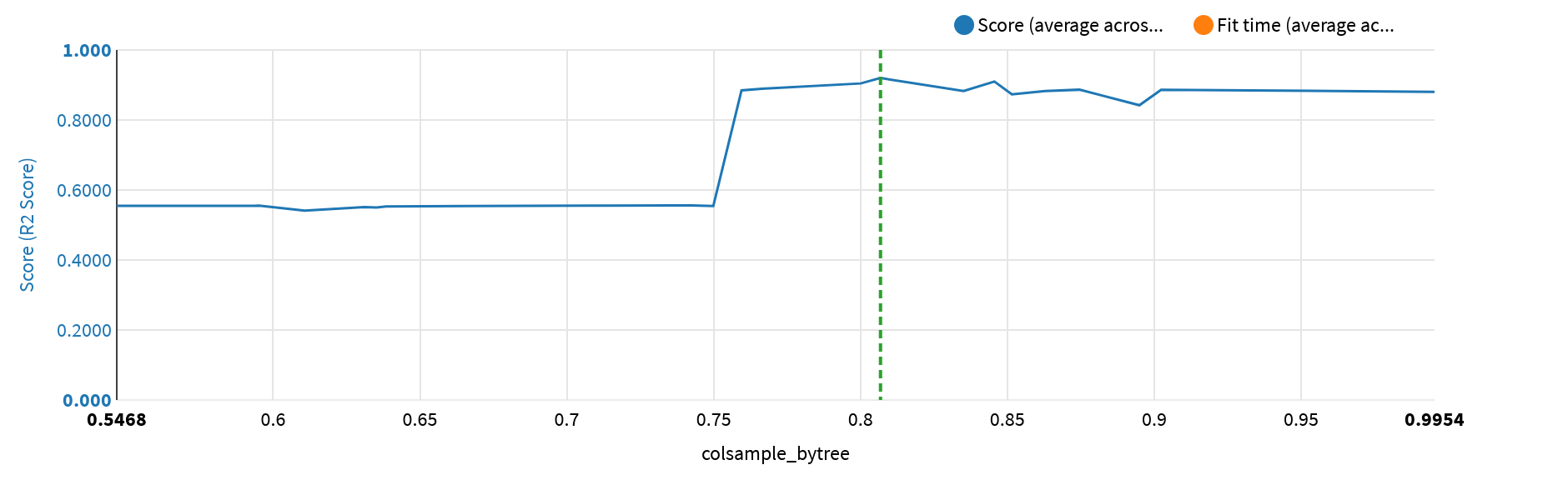
The optimal set of hyperparameters for the selected algorithm LightGBM is given in the table below:

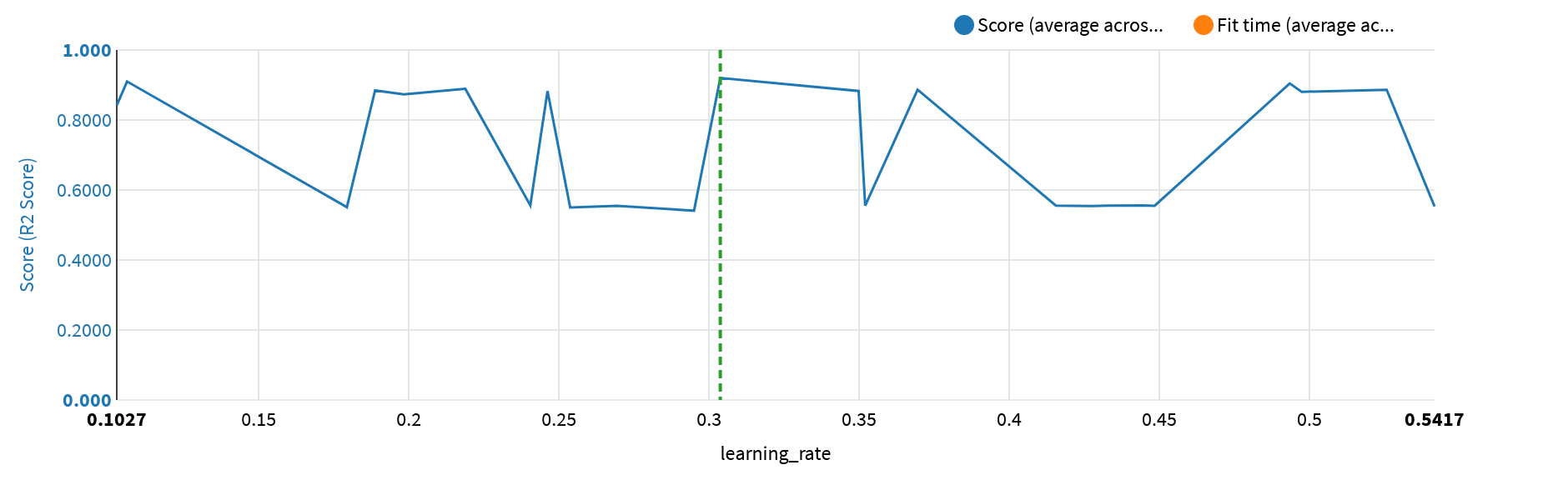
|  |  |
| --- | --- |
| Algorithm | LightGBM |
| Booster | gbdt |
| Actual number of trees | 35 |
| Maximum number of leaves | 170 |
| Learning rate | 0.3037391154768321 |
| Alpha (L1 regularization) | 0.7503401167613029 |
| Lambda (L2 regularization) | 0.22234662888213597 |
| Minimal gain to perform a split on a leaf | 0.6303148291660114 |
| Min sum of instance weight in a child | 0.013852804301969331 |
| Subsample ratio of the training instance | 0.75 |
| Columns subsample ratio for trees | 0.8068019527173328 |

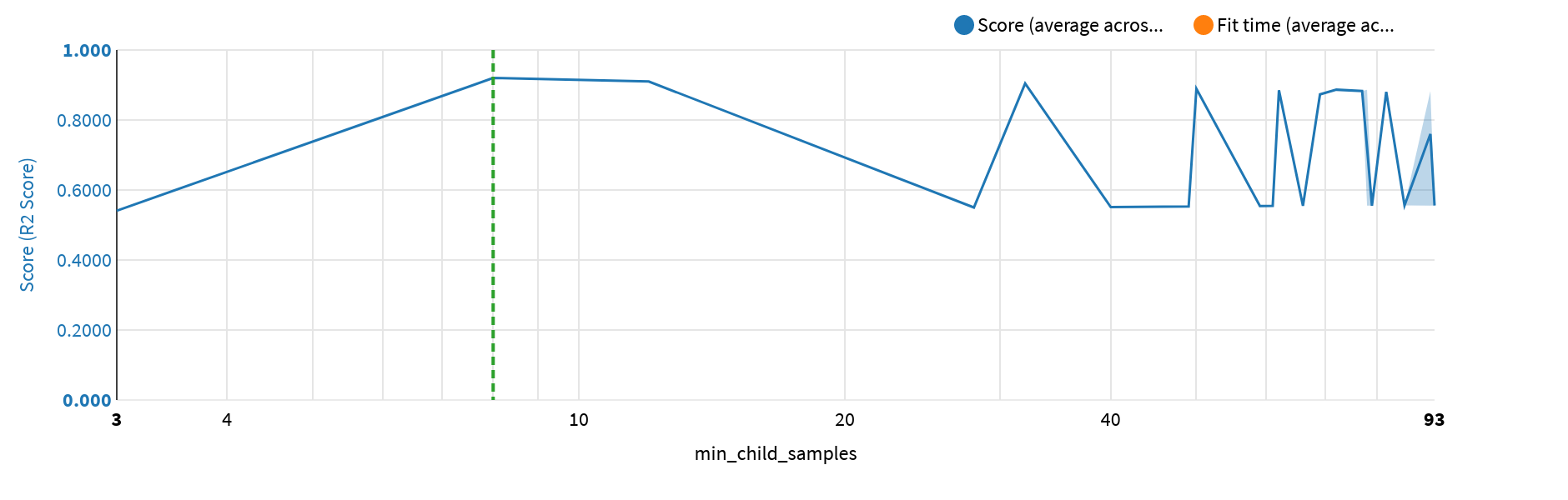
See section II.D.2.a) for detailed explanations on the algorithm and its hyperparameters.

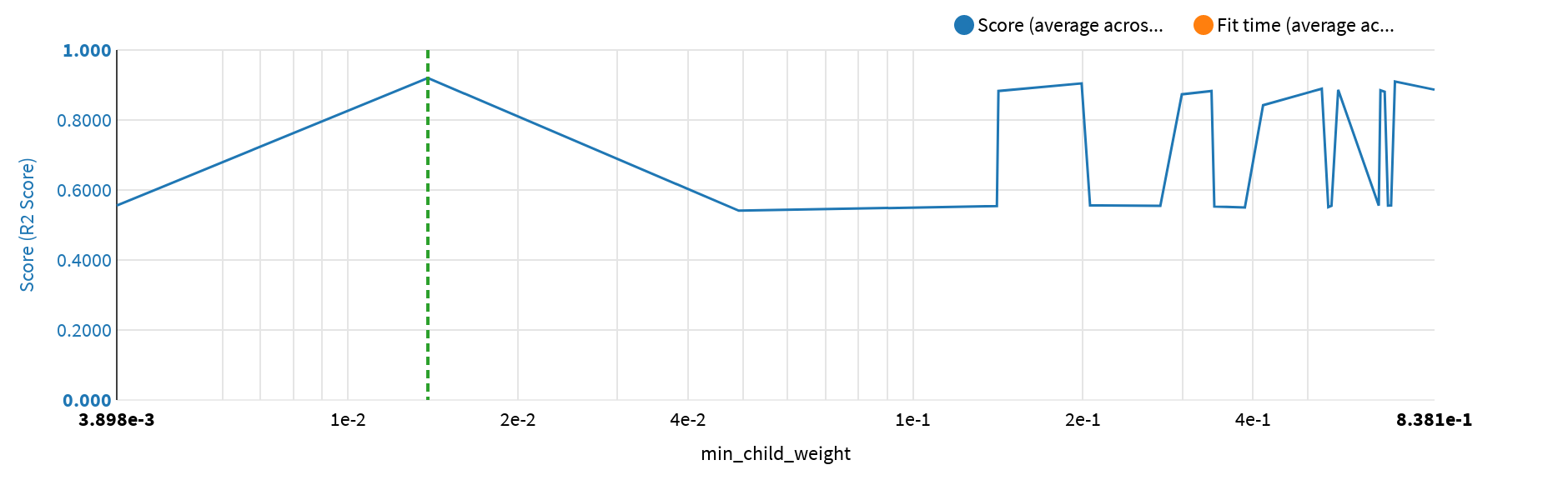
## Alternative Models

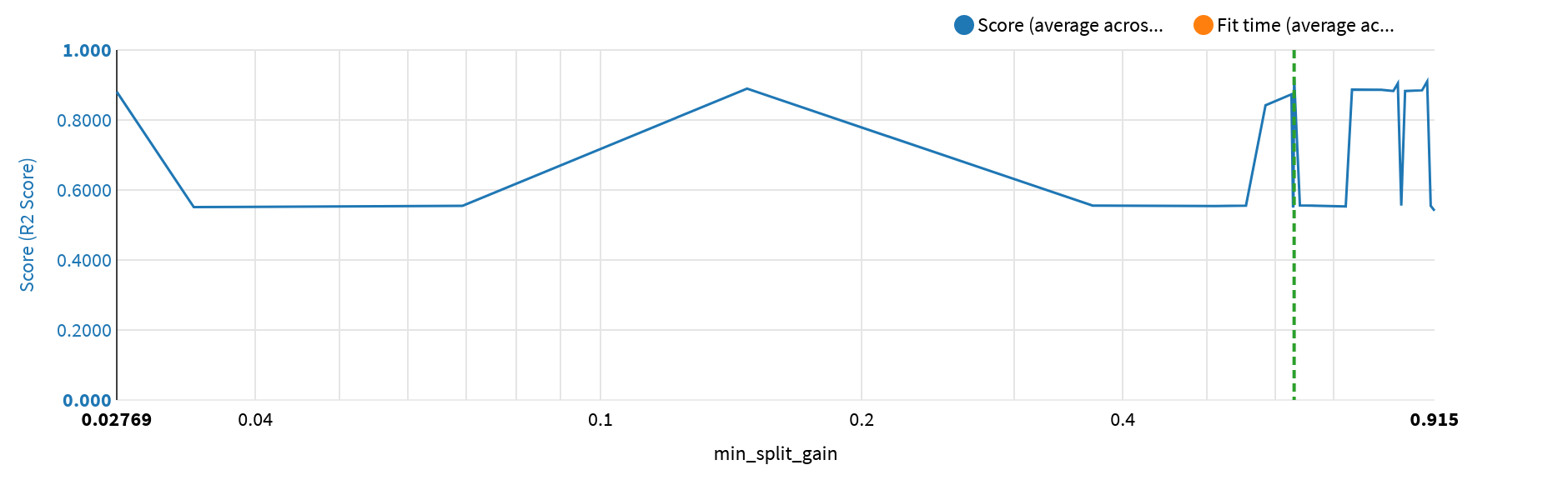
For the selected algorithm, the following other hyperparameter combinations were tried and led to lower performance. As an example, the plot below shows the evolution of the performance for each hyperparameter:

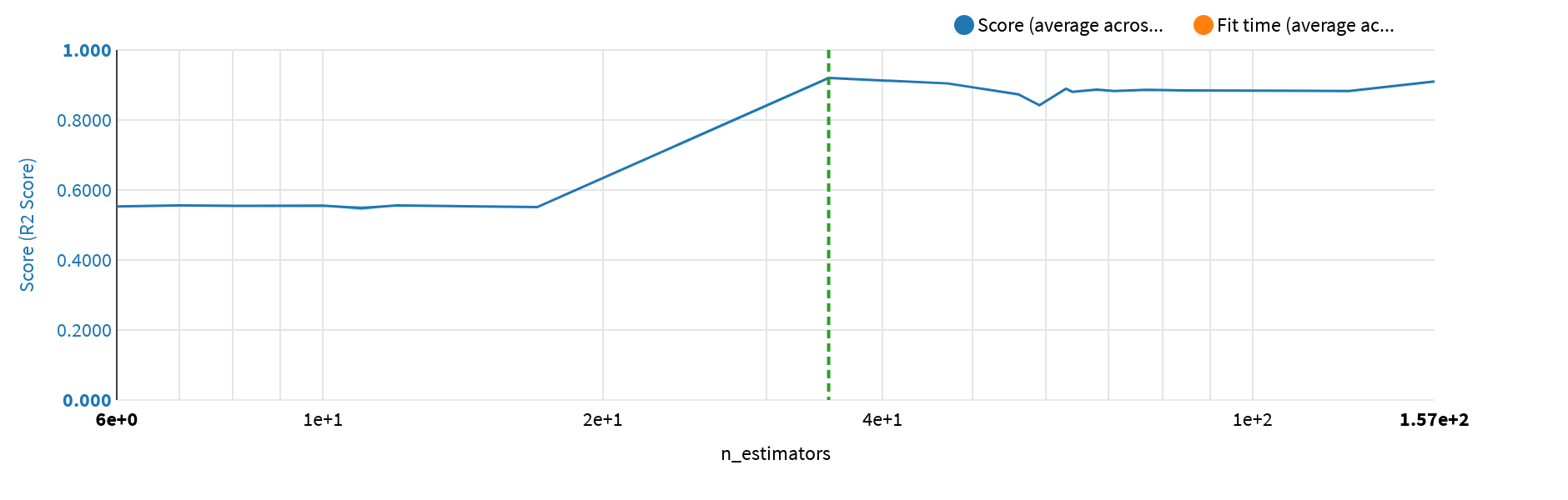


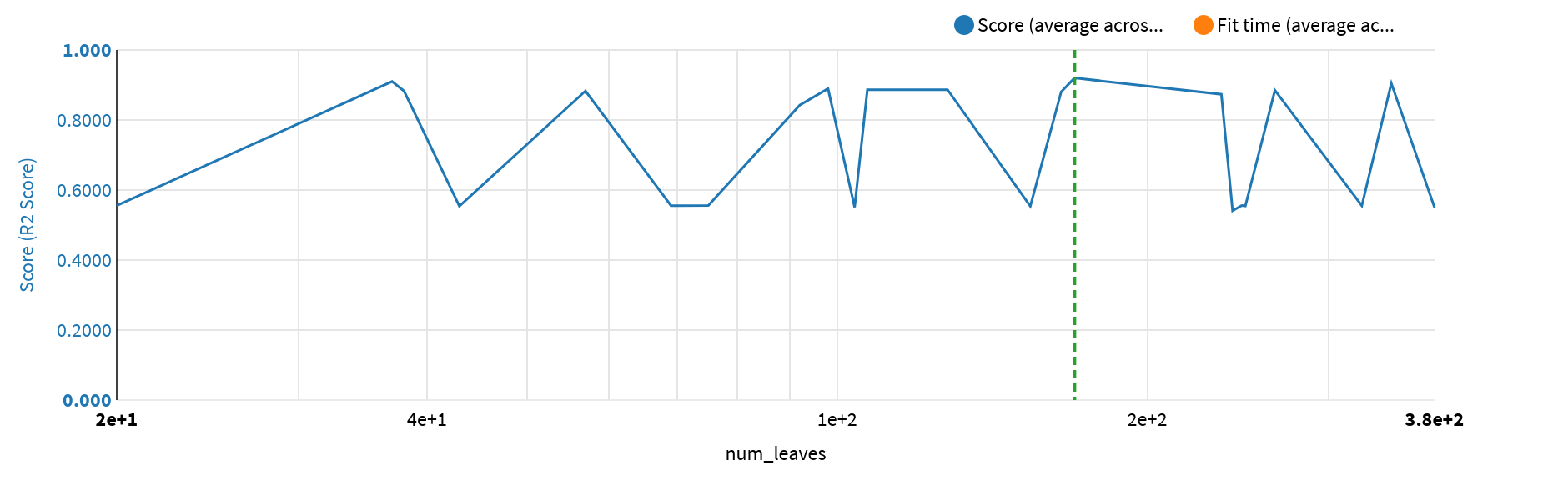


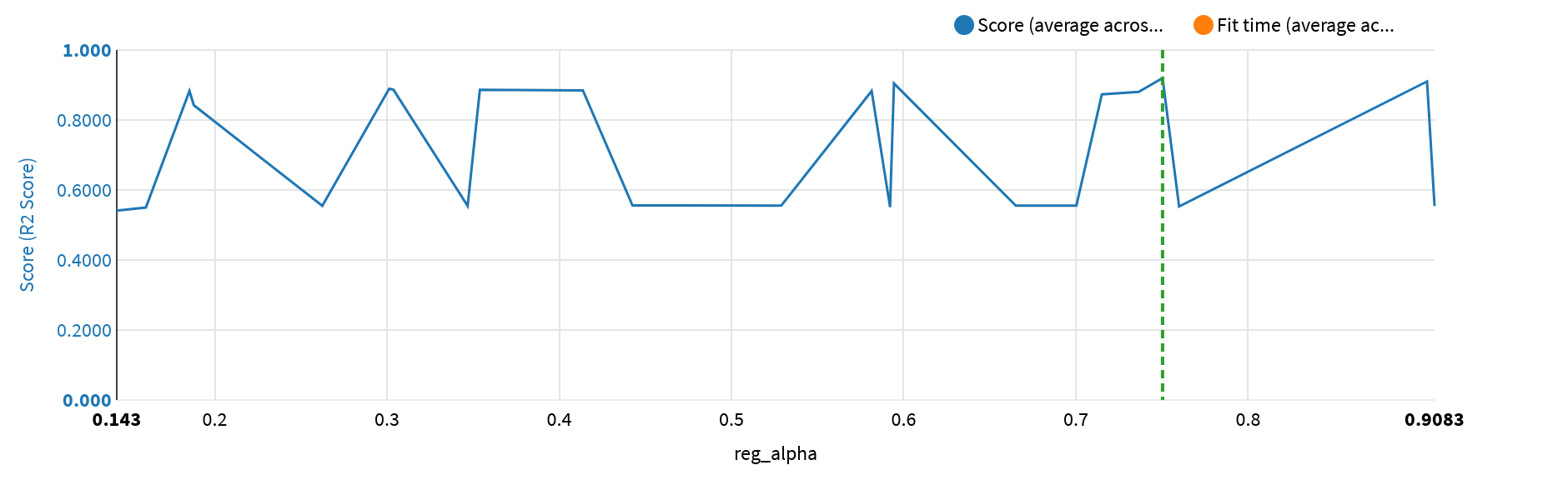


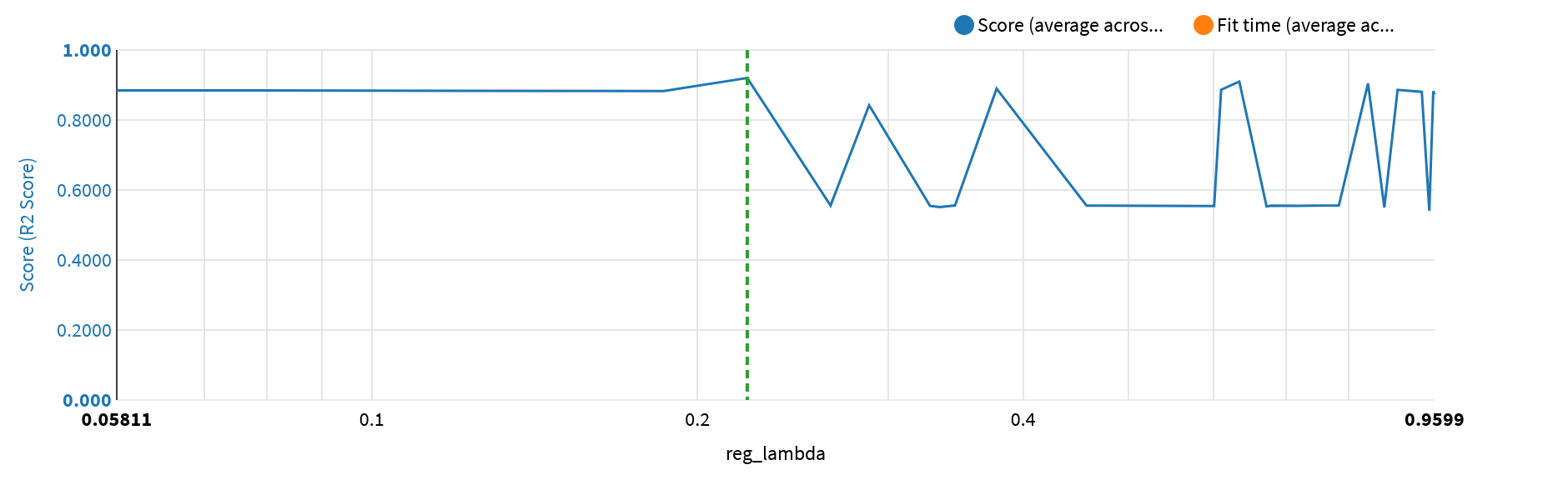












The table below lists all the performed trainings:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| colsample\_­bytree | learning\_­rate | min\_­child\_samples | min\_­child\_weight | min\_­split\_gain | n\_estimators | num\_­leaves | reg\_­alpha | reg\_­lambda | Score | Score StdDev | Fit Time | Fit Time StdDev | Score Time | Score Time StdDev |
| 0.5467607292345203 | 0.44842869126672547 | 61 | 0.5506560072282601 | 0.9057503615740888 | 8 | 154 | 0.2624024528483213 | 0.7191567925622773 | 0.554809 | 0.006845 | 0.462000 | 0.093014 | 0.015800 | 0.017520 |
| 0.5934641024693588 | 0.2692768113070123 | 66 | 0.2741295757693996 | 0.06928800089962439 | 11 | 249 | 0.3467971359200923 | 0.32805428660525804 | 0.554607 | 0.007014 | 0.658400 | 0.039813 | 0.010800 | 0.005706 |
| 0.595222762848161 | 0.4332323262849085 | 92 | 0.667661596223379 | 0.5547126822961741 | 7 | 20 | 0.700298420049446 | 0.26545033832019393 | 0.555604 | 0.005691 | 0.324200 | 0.113959 | 0.010200 | 0.004069 |
| 0.6107256674465726 | 0.29498492534883913 | 3 | 0.04918686346291431 | 0.9150254046173107 | 11 | 242 | 0.1430428482327608 | 0.9492866548689445 | 0.540885 | 0.008778 | 2.279600 | 0.251815 | 0.016000 | 0.017493 |
| 0.6310123375077908 | 0.1793419860772328 | 40 | 0.5435375953678605 | 0.033971386559311534 | 17 | 104 | 0.5921130348886767 | 0.3350268377072012 | 0.550990 | 0.007545 | 1.130800 | 0.106436 | 0.037400 | 0.016219 |
| 0.6352409524202889 | 0.25372641427544584 | 28 | 0.3868887935657251 | 0.6285011795397116 | 11 | 380 | 0.15994863542336868 | 0.8626475467499967 | 0.550010 | 0.007315 | 1.208200 | 0.430509 | 0.022600 | 0.018184 |
| 0.6384047529602773 | 0.5416881482730863 | 49 | 0.3415950356815954 | 0.7228576179347334 | 6 | 154 | 0.7599531776449456 | 0.6714027862126307 | 0.552925 | 0.007547 | 0.539200 | 0.064082 | 0.020800 | 0.021085 |
| 0.7079423866228542 | 0.41553488855518195 | 78 | 0.003898129138541196 | 0.3692290622594858 | 8 | 69 | 0.5260894484164497 | 0.678162693775239 | 0.555429 | 0.007020 | 0.414200 | 0.073912 | 0.018000 | 0.013251 |
| 0.7127073325722584 | 0.3520035195807186 | 79 | 0.6932944472489927 | 0.8376903141415869 | 10 | 323 | 0.6650964778876121 | 0.4577125841873495 | 0.555329 | 0.007010 | 0.488000 | 0.076718 | 0.021600 | 0.021814 |
| 0.7365412435615102 | 0.44440826194465444 | 93 | 0.7021272306924701 | 0.6291896895232066 | 7 | 247 | 0.5290150439332159 | 0.7830427916910186 | 0.555857 | 0.006069 | 0.356000 | 0.057921 | 0.015600 | 0.018424 |
| 0.7425732453726674 | 0.24044241859654217 | 86 | 0.20585343066724607 | 0.6400390792018065 | 12 | 75 | 0.4425383058157052 | 0.34594567321838143 | 0.555963 | 0.006296 | 0.588000 | 0.143188 | 0.022200 | 0.018170 |
| 0.7498685529912683 | 0.4275360742343617 | 59 | 0.14081677114249147 | 0.5112130608881936 | 8 | 43 | 0.9082882581540949 | 0.6005236842599005 | 0.554116 | 0.006818 | 0.474800 | 0.097602 | 0.016600 | 0.016317 |
| 0.7594793251929153 | 0.18873399244085057 | 62 | 0.6723118423049789 | 0.8851871400469946 | 84 | 266 | 0.4137199873026304 | 0.058114534564487785 | 0.884748 | 0.004916 | 3.804400 | 0.081303 | 0.116400 | 0.030572 |
| 0.766414731812736 | 0.2187553825409436 | 50 | 0.529308423041334 | 0.14754085807019113 | 63 | 98 | 0.3012482481431866 | 0.37797877486134757 | 0.889437 | 0.003195 | 3.495200 | 0.088935 | 0.068400 | 0.012339 |
| 0.8000488718771905 | 0.49343686611734205 | 32 | 0.19890140955981764 | 0.8300263664174055 | 47 | 345 | 0.5943665389420951 | 0.8331808533321858 | 0.904469 | 0.006593 | 4.531000 | 1.753492 | 0.056000 | 0.020030 |
| 0.8068019527173328 | 0.3037391154768321 | 8 | 0.013852804301969331 | 0.6303148291660114 | 35 | 170 | 0.7503401167613029 | 0.22234662888213597 | 0.919975 | 0.005881 | 7.095200 | 1.554637 | 0.071600 | 0.028563 |
| 0.8351094290801171 | 0.34982413068600626 | 77 | 0.1417132394368064 | 0.8201808869778754 | 71 | 38 | 0.18525094507204487 | 0.9573592236712607 | 0.882780 | 0.004074 | 2.673600 | 0.130101 | 0.087400 | 0.008452 |
| 0.8455800725203906 | 0.10610008429606824 | 12 | 0.7131746126577959 | 0.8972476309568498 | 157 | 37 | 0.903927199330352 | 0.63354860594173 | 0.910036 | 0.005313 | 6.553600 | 1.200695 | 0.136400 | 0.025928 |
| 0.851538932851199 | 0.19830943469006954 | 69 | 0.29923868146564925 | 0.625852826299868 | 56 | 236 | 0.7151044553551421 | 0.9599112777543806 | 0.873295 | 0.003822 | 1.983600 | 0.229063 | 0.077400 | 0.014472 |
| 0.8628255511068278 | 0.24618027256990874 | 92 | 0.33771037809916954 | 0.8462136572690205 | 127 | 57 | 0.5813986303244026 | 0.1861074404972819 | 0.882764 | 0.005410 | 4.386800 | 1.102034 | 0.171800 | 0.038665 |
| 0.8745288544743056 | 0.3694725975775619 | 72 | 0.8381132470025997 | 0.7349740509295771 | 68 | 128 | 0.3037163398343702 | 0.6095260451658263 | 0.886601 | 0.003841 | 2.878200 | 0.366742 | 0.078000 | 0.022441 |
| 0.8949476063032396 | 0.10267128035896109 | 92 | 0.41668783563142703 | 0.5842581281103214 | 59 | 92 | 0.18780840542641564 | 0.2881671478120357 | 0.842190 | 0.004766 | 1.983200 | 0.152758 | 0.038200 | 0.024277 |
| 0.9023501603423856 | 0.5257371661761103 | 78 | 0.5659025403295507 | 0.7944970919232228 | 77 | 107 | 0.3538981143103366 | 0.8872988393225 | 0.886295 | 0.008453 | 2.980600 | 0.523127 | 0.071800 | 0.021940 |
| 0.9954456221711241 | 0.497419542541366 | 82 | 0.6839694524022608 | 0.027689715150536642 | 64 | 165 | 0.7364419961241299 | 0.9342098037151884 | 0.880520 | 0.008678 | 2.436800 | 0.619732 | 0.082000 | 0.021707 |

# Selected Model Results

## Selected Model Metrics

The detailed metrics obtained on the test dataset are given below.

|  |  |  |
| --- | --- | --- |
| Explained Variance Score | Best possible score is 1.0, lower values are worse | 0.93083 |
| Mean Absolute Error (MAE) | Average of the absolute value of the regression error | 2.6391 |
| Mean Absolute Percentage Error | Average of the absolute value of the relative regression error | 5.40% |
| Mean Squared Error (MSE) | Average of the squares of the errors | 16.059 |

|  |  |  |
| --- | --- | --- |
| Root Mean Squared Error (RMSE) | Square root of the MSE | 4.0073 |
| Root Mean Squared Logarithmic Error (RMSLE) | Root of the average of the squares of the natural log of the regression error | 0.077753 |
| Pearson coefficient | Correlation coefficient between actual and predicted values. +1 = perfect correlation, 0 = no correlation, -1 = perfect anti-correlation | 0.96480 |
| R2 Score | (Coefficient of determination) regression score function | 0.93082 |

The ml assertions metrics are given below.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Criteria | Expected range | Expected valid ratio | Rows matching criteria | Rows dropped by the model | Valid ratio | Result |
|  | No assertions defined in the settings |  |  |  |  |  |  |

## Selected Model Performance Charts

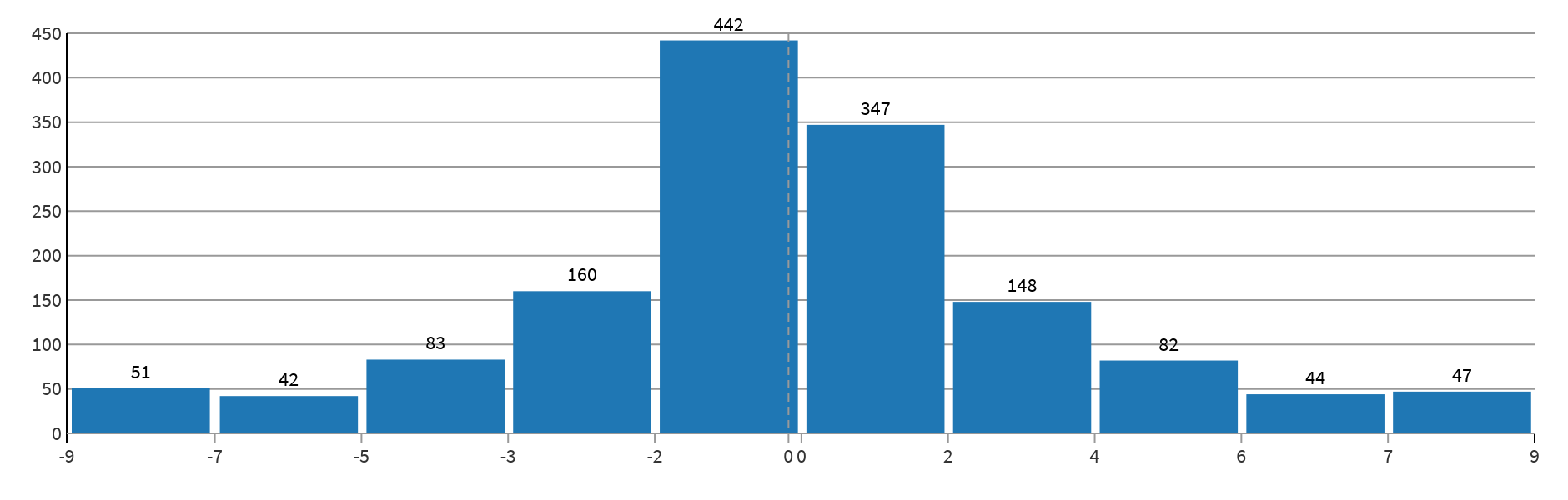
### Error Distribution

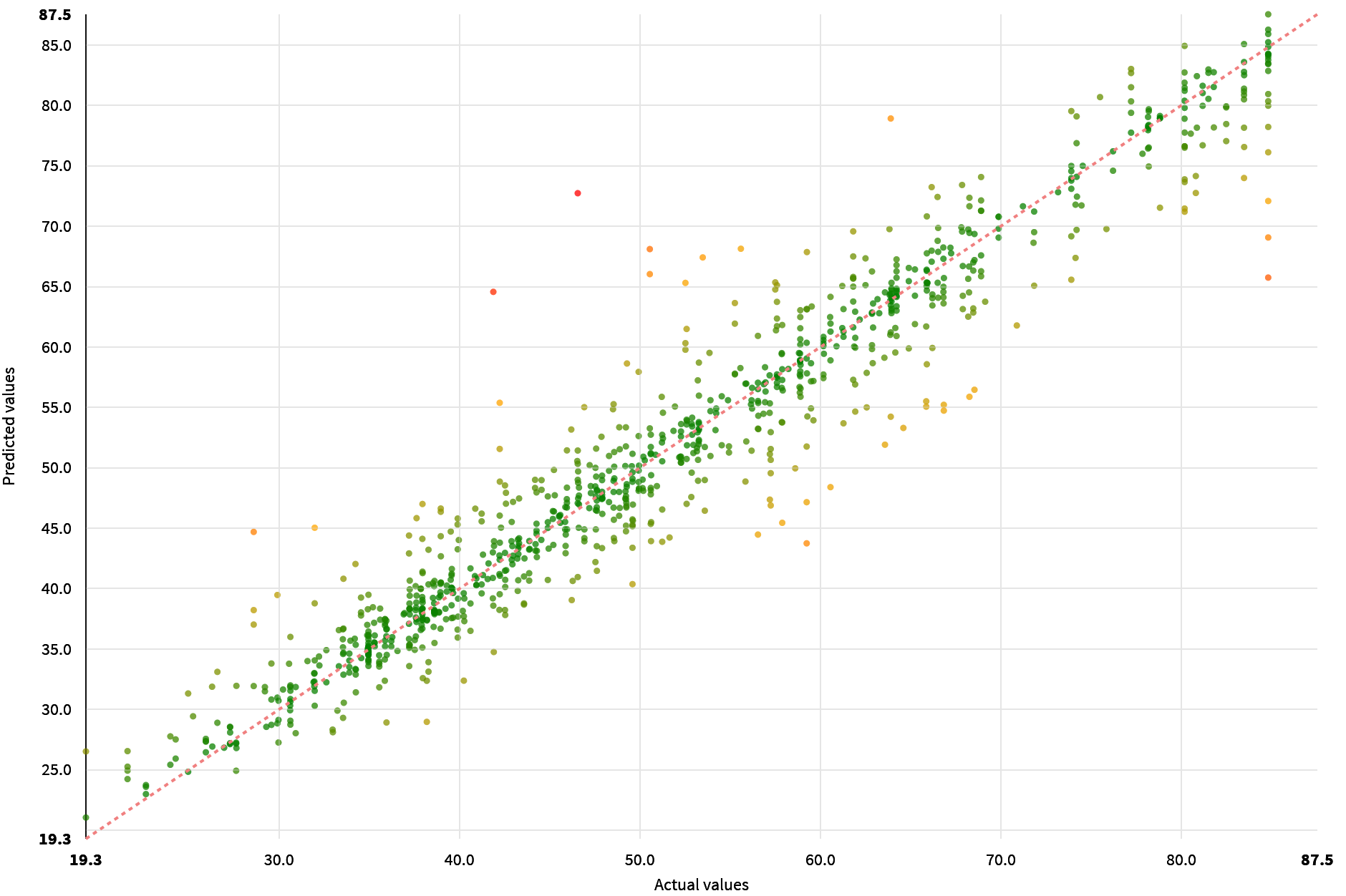
The error distribution table for this regression model is given below as a table with some statistics, as well as a histogram and as a scatter plot.

|  |  |
| --- | --- |
| Min. (raw) | -26.181 |
| Min. | -8.9234 |
| 25th perc. | -1.5108 |
| Median | -0.026075 |
| 75th perc. | 1.7219 |
| 90th parc. | 4.2982 |
| Max. | 9.2439 |
| Max. (raw) | 22.566 |
| Average. | 0.057907 |
| Standard deviation | 3.4378 |

The errors (the difference between predicted and actual values) should be centered around zero, and the distribution should be “narrow”, i.e., the spread of the error should be limited. More generally, the errors should be “normally” distributed around zero (the curve should look like a bell).

To reduce the effect of possible spurious outliers, the error distribution is winsorized (clipped) at the 2nd and 98th percentiles.





### Individual Prediction Explanations

Individual prediction explanations are feature importances specific to a given sample. They have been computed for the most extreme predictions.

|  |
| --- |
|  |

|  |
| --- |
| When the model is linear (logistic regression, OLS...), the explanation for one feature is simply the impact of the feature on the prediction with the mean feature value as a baseline:  coefficient \* (feature value - mean feature value)  As a generalization, the explanation is the difference between the prediction value and the average of prediction values obtained by replacing the feature value by values drawn from the test dataset. This method approximates Shapley values, trading off speed against both bias and variance.  For classification problems, the explanations are computed probability log-odd ratios:  log(p / (1 - p)) |

## Sensitivity Testing and Analysis

## Diagnostics

ML Diagnostics are designed to identify and help troubleshoot potential problems and suggest possible improvements at different stages of training and building machine learning models.

|  |  |
| --- | --- |
| Dataset sanity checks | Nothing to report |
| Modeling parameters | Nothing to report |
| Training speed | Nothing to report |
| Overfit detection | Nothing to report |
| Leakage detection | Nothing to report |
| Model check | Nothing to report |
| ML assertions | Nothing to report |
| Abnormal predictions detection | Nothing to report |

# Deployment and Monitoring

## Implementation Details

* The backend used by the model is: Python (in memory)
* The model can be found here: https://localhost:44000/projects/GEO/savedmodels/hjsVrlyp/p/S-GEO-hjsVrlyp-initial/#summary
* The name of the generated file is: Dataiku Model Documentation - Predict FloodHealthIndex (regression).docx
* The timing of the training was the following:

|  |  |
| --- | --- |
| Preprocessed in | 0.1s |
| Trained in | 73.1s |
| Loading train set | 0.0s |
| Loading test set | 0.0s |
| Collecting statistics | 0.0s |
| Preprocessing train set | 0.1s |
| Preprocessing test set | 0.0s |
| Hyperparameter searching | 70.0s |
| Fitting model | 2.4s |
| Saving model | 0.0s |
| Scoring model | 0.5s |

## Version Control

* The model was trained at 2022-10-08 21:55:02 (In the DSS server time zone).
* The model was trained with the following version of DSS: 11.0.3
* With the following code environment: DSS builtin environment