{{design.mltask.name}}

{{design.visual\_analysis.name}}

short line



|  |  |  |
| --- | --- | --- |
| **Version** | **Author** | **Date** |
| 1.0 | {{config.author.name}}  {{config.author.email}} | {{config.generation\_date.name}} |

Table of contents

[I. Executive Summary 4](#_Toc44339452)

# Executive Summary

{{if design.has\_multiple\_timeseries.value != Yes}}

A {{design.prediction\_type.name}} model was built using Dataiku DSS Visual Forecasting. Its goal is to forecast the {{design.target.name}} variable through time with a horizon of {{design.forecast\_horizon.in\_units.value}} ({{design.forecast\_horizon.time\_steps.value}} time steps of {{design.time\_step.value}}), and {{design.timeseries\_num\_external\_features.value}} external features. Using a dataset of {{result.train\_set.sample\_rows\_count.value}} rows, the process led to the selection of the {{result.chosen\_algorithm.name}} algorithm.

{{endif design.has\_multiple\_timeseries.value}}

{{if design.has\_multiple\_timeseries.value == Yes}}

{{if design.partitioned\_model.status == Yes}}

A {{design.prediction\_type.name}} model was built using Dataiku DSS Visual Forecasting. Its goal is to forecast the {{design.target.name}} variable through time with a horizon of {{design.forecast\_horizon.in\_units.value}} ({{design.forecast\_horizon.time\_steps.value}} time steps of {{design.time\_step.value}}), and {{design.timeseries\_num\_external\_features.value}} external features. Using a dataset of {{result.train\_set.sample\_rows\_count.value}} rows, the process led to the selection of the {{result.chosen\_algorithm.name}} algorithm.

{{endif design.partitioned\_model.status}}

{{if design.partitioned\_model.status == No}}

A {{design.prediction\_type.name}} model was built using Dataiku DSS Visual Forecasting. Its goal is to forecast the {{design.target.name}} variable through time for {{design.num\_timeseries.value}} series with a horizon of {{design.forecast\_horizon.in\_units.value}} ({{design.forecast\_horizon.time\_steps.value}} time steps of {{design.time\_step.value}}), and {{design.timeseries\_num\_external\_features.value}} external features. Using a dataset of {{result.train\_set.sample\_rows\_count.value}} rows, the process led to the selection of the {{result.chosen\_algorithm.name}} algorithm.

{{endif design.partitioned\_model.status}}

{{endif design.has\_multiple\_timeseries.value}}

## Methodology

{{if config.is\_saved\_model.value != Yes}}

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

{{endif config.is\_saved\_model.value}}

{{if config.is\_saved\_model.value == Yes}}

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:

{{endif config.is\_saved\_model.value}}

{{design.training\_and\_testing\_strategy.table}}

|  |  |
| --- | --- |
|  |  |
|  |  |

{{/design.training\_and\_testing\_strategy.table}}

{{if design.training\_and\_testing\_strategy.policy.value != Split the dataset}}

{{design.train\_set.image}}

{{design.test\_set.image}}

{{endif design.training\_and\_testing\_strategy.policy.value}}

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

{{if config.is\_saved\_model.value != Yes}}

Before being tested, the ML candidate models had been tuned to find the best combination of hyperparameters according to the {{design.test\_metrics.name}} metric. This optimal hyperparameter search, based on assessing performance on a validation set, was done using the following methodology:

{{endif config.is\_saved\_model.value}}

{{if config.is\_saved\_model.value == Yes}}

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

{{endif config.is\_saved\_model.value}}

{{design.hyperparameter\_search\_strategy.table}}

|  |  |
| --- | --- |
|  |  |
|  |  |

{{/design.hyperparameter\_search\_strategy.table}}

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

## Results

{{if design.k\_fold\_cross\_testing.value== Yes}} }}

The {{result.chosen\_algorithm.name}} algorithm was selected. The evaluation metric used to tune the hyperparameters was {{design.test\_metrics.name}} computed on the validation dataset. After the best hyperparameter combination was found, the same metric was also computed on the test datasets. The final value was {{result.test\_metrics.value}}.

{{endif design.k\_fold\_cross\_testing.value}}

{{if design.k\_fold\_cross\_testing.value!= Yes }}

The {{result.chosen\_algorithm.name}} algorithm was selected. The evaluation metric used to tune the hyperparameters was {{design.test\_metrics.name}} 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 {{result.test\_metrics.value}}.

{{endif design.k\_fold\_cross\_testing.value}}

# Methodology

This section deals with the methodological details:

{{if dataset.prepare\_steps.status != No}}

* *Data Preparation* steps may first be applied to generate the initial set of features used in the process and may also transform the target.

{{endif dataset.prepare\_steps.status }}

{{if design.partitioned\_model.status == Yes}}

* The *Problem Definition* consists of selecting the target (**{{design.target.name}}**) and the type of problem ({{design.prediction\_type.name}}) together with the segmentation.

{{endif design.partitioned\_model.status}}

{{if design.partitioned\_model.status != Yes}}

* The *Problem Definition* consists of selecting the target (**{{design.target.name}}**) and the type of problem ({{design.prediction\_type.name}}).

{{endif design.partitioned\_model.status}}

* *Data Ingestion* resamples the time series if necessary, and analyzes each external feature in order to maximize its prediction potential.

{{if config.is\_saved\_model.value == Yes}}

* *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.

{{endif config.is\_saved\_model.value}}

{{if config.is\_saved\_model.value != Yes}}

* *Model and Feature Tuning* describes the tested algorithms and the way to find the best hyperparameter set for each of them.
* The *Model Evaluation and Selection* strategy indicates how to compute the metrics that allow for comparison between the best-tuned algorithms so that the user can select the best algorithm according to one of the computed metrics (here {{result.chosen\_algorithm.name}}).

{{endif config.is\_saved\_model.value}}

{{if dataset.prepare\_steps.status != No}}

## Data Preparation

The following data preparation steps are applied to the initial dataset in order to produce the Machine Learning dataset with {{design.features\_count.value}} features and {{result.train\_set.sample\_rows\_count.value}} samples:

{{dataset.prepare\_steps.table}}

|  |
| --- |
|  |
|  |

{{/dataset.prepare\_steps.table}}

{{endif dataset.prepare\_steps.status }}

## Problem Definition

{{if design.partitioned\_model.status == Yes}}

### Target Selection

{{endif design.partitioned\_model.status}}

{{if design.has\_multiple\_timeseries.value != Yes}}

A {{design.prediction\_type.name}} model was built using Dataiku DSS. Its goal is to forecast the {{design.target.name}} variable through time, defined by the following columns in the input dataset:

{{endif design.has\_multiple\_timeseries.value}}

{{if design.has\_multiple\_timeseries.value == Yes}}

{{if design.partitioned\_model.status == No}}

A {{design.prediction\_type.name}} model was built using Dataiku DSS. Its goal is to forecast the {{design.target.name}} variable through time for {{design.num\_timeseries.value}} series, defined by the following columns in the input dataset:

{{endif design.partitioned\_model.status}}

{{if design.partitioned\_model.status == Yes}}

A {{design.prediction\_type.name}} model was built using Dataiku DSS. Its goal is to forecast the {{design.target.name}} variable through time, defined by the following columns in the input dataset:

{{endif design.partitioned\_model.status}}

{{endif design.has\_multiple\_timeseries.value}}

{{design.timeseries\_general\_settings.table}}

|  |  |
| --- | --- |
|  |  |
|  |  |

{{/design.timeseries\_general\_settings.table}}

The parameters defining the forecasting are as follows:

{{design.timeseries\_forecasting.table}}

|  |  |
| --- | --- |
|  |  |
|  |  |

{{/design.timeseries\_forecasting.table}}

{{design.forecasting\_parameters.image}}

{{if design.partitioned\_model.status == Yes}}

### Segmentation

One independent model is made for each partition listed below.

{{result.partitioned.summary.image}}

{{endif design.partitioned\_model.status}}

## Data Ingestion

During the data ingestion phase, the time series is resampled to match the requested sampling frequency and start/end points defined above. The table below summarizes the interpolation and extrapolation methods used.

{{result.timeseries\_resampling.table}}

|  |  |
| --- | --- |
|  |  |
|  |  |

{{/result.timeseries\_resampling.table}}

The external 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 external feature (status *Input*). Also, it shows the processing applied to columns that define the different time series (status *Timeseries identifier*) is any, and the columns for the target (status *Target*) and time variable (status *Time*).

{{design.input\_feature.table}}

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

{{/design.input\_feature.table}}

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

## Model and Feature Tuning

{{if config.is\_saved\_model.value != Yes}}

### Tested Algorithms

A selection of algorithms (candidate models) was then trained on the Machine Learning dataset, with various combinations of hyperparameters. The section below details the tested algorithms and the space of hyperparameters for each of them. It begins with the selected algorithm and its hyperparameter selection and continues with the other tested algorithms.

#### Selected Model

The {{result.chosen\_algorithm.name}} algorithm has been finally selected.

|  |
| --- |
| {{design.chosen\_algorithm\_search\_strategy.text}} |

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

{{design.chosen\_algorithm\_search\_strategy.table}}

|  |  |
| --- | --- |
|  |  |
|  |  |

{{/design.chosen\_algorithm\_search\_strategy.table}}

#### Alternative Models

Other algorithms are also tested. They are listed below, along with their settings:

{{design.other\_algorithms\_search\_strategy.table}}

|  |  |
| --- | --- |
|  |  |
|  |  |

{{/design.other\_algorithms\_search\_strategy.table}}

{{endif config.is\_saved\_model.value}}

{{if config.is\_saved\_model.value == Yes}}

### Tested Algorithm

The {{result.chosen\_algorithm.name}} algorithm has been tested.

|  |
| --- |
| {{design.chosen\_algorithm\_search\_strategy.text}} |

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

{{design.chosen\_algorithm\_search\_strategy.table}}

|  |  |
| --- | --- |
|  |  |
|  |  |

{{/design.chosen\_algorithm\_search\_strategy.table}}

{{endif config.is\_saved\_model.value}}

### Hyperparameter Search

{{if design.cross\_validation\_strategy.value == K-fold cross-test}}

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 ({{design.test\_metrics.name}}) computed on the validation datasets.

{{endif design.cross\_validation\_strategy.value}}

{{if design.cross\_validation\_strategy.value!= K-fold cross-test}}

{{if design.cross\_validation\_strategy.value == Time-based K-fold (with overlap)}}

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 ({{design.test\_metrics.name}}) computed on the validation datasets.

{{endif design.cross\_validation\_strategy.value}}

{{if design.cross\_validation\_strategy.value != Time-based K-fold (with overlap)}}

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 ({{design.test\_metrics.name}}) computed on the validation dataset.

{{endif design.cross\_validation\_strategy.value}}

{{endif design.cross\_validation\_strategy.value}}

{{if config.is\_saved\_model.value != Yes}}

The actual search settings for all the tested algorithms, including the selected one, are the following:

{{endif config.is\_saved\_model.value}}

{{if config.is\_saved\_model.value == Yes}}

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

{{endif config.is\_saved\_model.value}}

{{design.hyperparameter\_search\_strategy.table}}

|  |  |
| --- | --- |
|  |  |
|  |  |

{{/design.hyperparameter\_search\_strategy.table}}

|  |
| --- |
| **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. |

Illustration:

{{design.hyperparameter\_search\_strategy.image}}

{{if design.k\_fold\_cross\_testing.value != Yes}}

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

{{endif design.k\_fold\_cross\_testing.value}}

## 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.
* Performing a time-based K-fold cross-test. 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.

{{design.training\_and\_testing\_strategy.table}}

|  |  |
| --- | --- |
|  |  |
|  |  |

{{/design.training\_and\_testing\_strategy.table}}

{{if design.training\_and\_testing\_strategy.policy.value != Split the dataset}}

{{design.train\_set.image}}

{{design.test\_set.image}}

{{endif design.training\_and\_testing\_strategy.policy.value}}

{{if design.training\_and\_testing\_strategy.policy.value == Split the dataset}}

Illustration:

{{design.sampling.image}}

{{design.splitting.image}}

{{endif design.training\_and\_testing\_strategy.policy.value}}

|  |
| --- |
| **Legend**   * Policy:   + *Split the dataset:* Split a subset of the dataset. * *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 {{result.train\_set.sample\_rows\_count.value}} 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:* Time series forecasting models use time-based ordering. This 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. * *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.

{{if design.partitioned\_model.status != Yes}}

{{if config.is\_saved\_model.value != Yes}}

## Selected Model

{{result.chosen\_algorithm.name}} was finally selected by the user with the optimal set of hyperparameters given in the table below:

{{endif config.is\_saved\_model.value}}

{{if config.is\_saved\_model.value == Yes}}

{{if result.hyperparameter\_search.status != No}}

## Selected Model

{{endif result.hyperparameter\_search.status}}

The optimal set of hyperparameters for the selected algorithm {{result.chosen\_algorithm.name}} is given in the table below:

{{endif config.is\_saved\_model.value}}

{{result.chosen\_algorithm\_details.table}}

|  |  |
| --- | --- |
|  |  |
|  |  |

{{/result.chosen\_algorithm\_details.table}}

{{if result.chosen\_algorithm.name == AutoARIMA}}

ARIMA Orders:

{{if design.has\_multiple\_timeseries.value == Yes}}

{{result.autoarima\_orders\_multi.table}}

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

{{/result.autoarima\_orders\_multi.table}}

{{endif design.has\_multiple\_timeseries.value}}

{{if design.has\_multiple\_timeseries.value == No}}

{{result.autoarima\_orders\_single.table}}

|  |  |
| --- | --- |
|  |  |
|  |  |

{{/result.autoarima\_orders\_single.table}}

{{endif design.has\_multiple\_timeseries.value}}

{{endif result.chosen\_algorithm.name}}

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

{{if config.is\_saved\_model.value == Yes}}

{{if result.hyperparameter\_search.status != No}}

## 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:

{{result.hyperparameter\_search.plot}}

The table below lists all the performed trainings:

{{result.hyperparameter\_search.table}}

|  |  |
| --- | --- |
| Header | Header |
| Content | Content |

{{/result.hyperparameter\_search.table}}

{{endif result.hyperparameter\_search.status}}

{{endif config.is\_saved\_model.value}}

{{if config.is\_saved\_model.value != Yes}}

## Alternative Models

{{if result.hyperparameter\_search.status != No}}

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:

{{result.hyperparameter\_search.plot}}

The table below lists all the performed trainings:

{{result.hyperparameter\_search.table}}

|  |  |
| --- | --- |
| Header | Header |
| Content | Content |

{{/result.hyperparameter\_search.table}}

{{endif result.hyperparameter\_search.status}}

{{endif design.partitioned\_model.status}}

The selected algorithm was compared with other algorithms. The table below gives the performance obtained with the combination of hyperparameters that optimizes the {{design.test\_metrics.name}} metric:

{{result.leaderboard.image}}

Complete performance results obtained with the other evaluated metrics are given below:

|  |  |
| --- | --- |
| Mean Absolute Scaled Error | {{result.leaderboard.mase.image}} |
| Mean Absolute Percentage Error | {{result.leaderboard.mape.image}} |
| Symmetric Mean Absolute Percentage Error | {{result.leaderboard.smape.image}} |
| Mean Absolute Error | {{result.leaderboard.mae.image}} |
| Mean Absolute Quantile Loss | {{result.leaderboard.mean\_absolute\_quantile\_loss.image}} |
| Mean Weighted Quantile Loss | {{result.leaderboard.mean\_weighted\_quantile\_loss.image}} |
| Mean Squared Error | {{result.leaderboard.mse.image}} |
| Root Mean Squared Error | {{result.leaderboard.rmse.image}} |
| Mean Scaled Interval Score | {{result.leaderboard.msis.image}} |
| Normalized Deviation | {{result.leaderboard.nd.image}} |

{{endif config.is\_saved\_model.value}}

# Selected Model Results

{{if design.has\_multiple\_timeseries.value == No}}

## Selected Model Metrics

{{if design.k\_fold\_cross\_testing.value== Yes}}

The metrics given below are obtained on the test datasets (folds). As K-fold cross-testing was chosen, the average over the test datasets (folds) is given as well as a confidence interval.

{{endif design.k\_fold\_cross\_testing.value}}

{{if design.k\_fold\_cross\_testing.value!= Yes}}

The metrics given below are obtained on the test dataset.

{{endif design.k\_fold\_cross\_testing.value}}

{{result.detailed\_metrics.table}}

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

{{/result.detailed\_metrics.table}}

{{endif design.has\_multiple\_timeseries.value}}

{{if design.has\_multiple\_timeseries.value == Yes}}

## Selected Model Aggregated Metrics

{{if design.k\_fold\_cross\_testing.value== Yes}}

The metrics given below are obtained on the test datasets (folds) and aggregated across all the time series. As K-fold cross-testing was chosen, the average over the test datasets (folds) is given as well as a confidence interval.

{{endif design.k\_fold\_cross\_testing.value}}

{{if design.k\_fold\_cross\_testing.value!= Yes}}

The metrics given below are obtained on the test dataset and aggregated across all the time series.

{{endif design.k\_fold\_cross\_testing.value}}

{{result.detailed\_metrics.table}}

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

{{/result.detailed\_metrics.table}}

{{if design.partitioned\_model.status == No}}

## Selected Model Per Time Series Metrics

{{if design.k\_fold\_cross\_testing.value== Yes}}

The metrics given below are obtained on the test datasets (folds) and given per time series. As K-fold cross-testing was chosen, the average over the test datasets (folds) is given as well as a confidence interval.

{{endif design.k\_fold\_cross\_testing.value}}

{{if design.k\_fold\_cross\_testing.value!= Yes}}

The metrics given below are obtained on the test dataset and given per time series.

{{endif design.k\_fold\_cross\_testing.value}}

{{result.per\_timeseries\_metrics.table}}

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

{{/result.per\_timeseries\_metrics.table}}

{{endif design.has\_multiple\_timeseries.value}}

## Selected Model Performance Charts

{{if design.has\_multiple\_timeseries.value == No}}

### Forecast Chart

{{result.timeseries\_single\_forecast.plot}}

{{endif design.has\_multiple\_timeseries.value}}

{{if design.has\_multiple\_timeseries.value == Yes}}

### Forecast Charts

For multiple time series, up to 5 forecast charts are shown.

{{result.timeseries\_multi\_forecast.plot}}

{{endif design.has\_multiple\_timeseries.value}}

{{endif design.partitioned\_model.status}}

## Scoring new data

When using this model to score new data, the input dataset must contain {{design.timeseries\_scoring\_data\_length.value}}.

{{if design.timeseries\_num\_external\_features.value != 0}}

The input dataset must contain at least {{design.forecast\_horizon.in\_units.value}} of future values of external features.

{{endif design.timeseries\_num\_external\_features.value}}

{{if design.timeseries\_algorithm\_can\_score\_new\_series.value == Yes}}

This model can forecast values for new time series that it has not been trained on.

{{endif design.timeseries\_algorithm\_can\_score\_new\_series.value}}

{{if design.timeseries\_algorithm\_can\_score\_new\_series.value == No}}

{{if design.partitioned\_model.status == No}}

This model can only forecast values for series it has been trained on. If a scoring is requested on a series other than one of the {{design.num\_timeseries.value}} series it has been trained on, the scoring recipe will fail.

{{endif design.partitioned\_model.status}}

{{if design.partitioned\_model.status == Yes}}

This model can only forecast values for series it has been trained on. If a scoring is requested on a series other than one of the series it has been trained on, the scoring recipe will fail.

{{endif design.partitioned\_model.status}}

{{endif design.timeseries\_algorithm\_can\_score\_new\_series.value}}

## 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.

{{result.diagnostics.table}}

|  |  |
| --- | --- |
|  |  |
|  |  |

{{/result.diagnostics.table}}

# Deployment and Monitoring

## Implementation Details

* The backend used by the model is: {{design.backend.name}}
* The model can be found here: {{config.project.link}}
* The name of the generated file is: {{config.output\_file.name}}
* The timing of the training was the following:

{{result.timings.table}}

|  |  |
| --- | --- |
|  |  |
|  |  |

{{/result.timings.table}}

## Version Control

* The model was trained at {{result.training\_date.name}} (In the DSS server time zone).
* The model was trained with the following version of DSS: {{config.dss.version}}
* With the following code environment: {{config.environment.name}}