# Jedox AIssisted™ Planning-Services Documentation

## Interpolation Service:

**Name:** interpolationJedox

**Aim:** To replace missing values with meaningful values

**Use Case:** There may be cases that data contains missing values, for example, when budget planning, there might be historic planning data with missing values. To avoid errors due to this missing information, interpolation is used.

**Description:** The interpolation service takes numeric values and the type of algorithm as inputs. The interpolation algorithms used in this AIssisted service are divided into 3 categories: Simple, Intermediate and Advanced. The simple algorithms use simple techniques such as last observation carried forward, moving average with/without weights, random sampling, linear approximation, spline etc., for replacing missing values. The intermediate methods consider the seasonality of the data while replacing missing values. The advanced methods include Kalman smoothing and kNN missing value imputation. **NOTE:** The advanced methods namely (Kalman smoothing and kNN imputation) should be used with large number of input values only, preferably >12 values.

**Call:**

https://${AIssisted_Planning_Server_IPorName}:12800/api/interpolationJedox/v1.0.0

https://${AIssisted_Planning_Server_IPorName}:12800/api/interpolationJedox/v1.1.1

https://${AIssisted_Planning_Server_IPorName}:12800/api/interpolationJedox/v1.1.2

TABLE 1: Group Types for Interpolation service

| GROUP | VALUE |
|---|---|
| Simple | 0 |
| Seasonal | 1 |
| Advanced | 2 |

TABLE 2: Group, Algorithm and Subtype combinations for interpolation service Version (v1.1.0).

| GROUP | ALGORITHM | SUB-TYPE | DESCRIPTION |
|---|---|---|---|
| 0 | 0 | Not Applicable | Last observation carried forward(LOCF) |
| | 1 | Not Applicable | Linear approximation |
| | 2 | Not Applicable | Spline |
| | 3 | Not Applicable | Random Sampling |
| | 4 | 0 | Moving average with equal weights for all observations |
| | | 1 | Moving average with linear weights |
| | | 2 | Moving average with exponential weights |
| | 5 | 0 | Mean |
| | | 1 | Mode |
| | | 2 | Median |
| 1 | 0 | 0 | Seasonal decomposition with LOCF |
| 1 | 0 | 1 | Seasonal decomposition with mean |
| | | 2 | Seasonal decomposition with linear approximation |
| | 1 | 0 | Seasonal split with LOCF |
| | | 1 | Seasonal split with mean |
| | | 2 | Seasonal split with linear approximation |
| 2 | 0 | 0 | Kalman smoothing with basic ARIMA model |
| | | 1 | Kalman smoothing with basic state space model |
| | 1 | Not Applicable | k-nearest neighbors with median |
| | 2 | Not Applicable | Linear regression model |

**Input:**

1. *inputValues* string= comma separated string of input values
2. *period(default=12)* numeric= type of time; yearly=1, monthly=12, weekly=52, daily=365
3. *numInputs* numeric = type of input; Single input for interpolating = 0/1, Multiple inputs for interpolating > 1
4. *algGroup* numeric = algorithm group; Simple = 0, Intermediate = 1, Advanced = 2
5. *alg* numeric = algorithm number in each group; Group 0 = 0 to 5, Group 1 = 0,1, Group 2 = 0,1,2
6. *algSubType* numeric = to set the metric to be used as algorithm parameter; Group 0 Alg 4/5 SubType = 0,1,2 Group 1 Alg 0/1 SubType = 0,1,2 Group 2 Alg 0 SubType = 0,1
7. *outputType* numeric = interpolated values=0, interpolated values and position of missing values = 1. The positions are identified as 1 for 0 in input, 2 for NA in input and 0 otherwise.

**Input V1.1.1, V1.1.2:**

1. *InputValues* string= comma separated string of input values
2. *Period(default=12)* numeric= type of time; yearly=1, monthly=12, weekly=52, daily=365
3. *NumInputs* numeric = type of input; Single input for interpolating = 0/1, Multiple inputs for interpolating > 1
4. *AlgGroup* numeric = algorithm group; Simple = 0, Intermediate = 1, Advanced = 2
5. *AlgorithmType* numeric = algorithm number in each group; Group 0 = 0 to 5, Group 1 = 0,1, Group 2 = 0,1,2
6. *AlgSubType* numeric = to set the metric to be used as algorithm parameter; Group 0 Alg 4/5 SubType = 0,1,2 Group 1 Alg 0/1 SubType = 0,1,2 Group 2 Alg 0 SubType = 0,1
7. *OutputType* numeric = interpolated values=0, interpolated values and position of missing values = 1. The positions are identified as 1 for 0 in input, 2 for NA in input and 0 otherwise.
8. *user* **(V1.1.1 only)** string = AIssisted username e.g., john@aissistedplanning.cloud.jedox.com
9. *apikey* **(V1.1.1 only)** string = AIssisted apikey
10. *AIssistedUser* **(V1.1.2 only)** string = AIssisted username e.g., john@aissistedplanning.cloud.jedox.com
11. *AIssistedKey* **(V1.1.2 only)** string = AIssisted apikey

**Output:** outputParameters.values[…] contains the following in the order as mentioned.

0. *Values* = input values with missing values replaced by interpolated values.
1. *Positions* = positions of missing values (only with outputType = 1)

## Outlier Detection Service:

**Name:** outlierJedox

**Aim:** To detect and replace extreme values with meaningful values.

**Use Case:** The outlier detection methods are used for automatic detection of values that do not fit the general behavior of the other values in the given set. In simple terms, when a user enters data, it is possible to add an accidental zero, or any extra number. This results in deviation from the accurate forecasts when this in accurate data is used for planning and forecasting. In order to avoid the manual overhead of ensuring the correctness of data, the outlier detection service can be used.

**Description:** The outlier detection service takes values and the type of algorithm as inputs along with replacement specific parameters. By default, the replacement is set to linear approximation for the outlier values. The user can decide against replacement or choose other replacement algorithms from the list of algorithms in the interpolation service.

**Call:**

https://${AIssisted_Planning_Server_IPorName}:12800/api/outlierJedox/v1.2.0

https://${AIssisted_Planning_Server_IPorName}:12800/api/outlierJedox/v1.3.0

https://${AIssisted_Planning_Server_IPorName}:12800/api/outlierJedox/v1.3.1

https://${AIssisted_Planning_Server_IPorName}:12800/api/outlierJedox/v1.3.2

TABLE 3: Algorithms for Time Series Outlier Detection

| Algorithm | Short Description |
|---|---|
| IO | Innovational outlier detection |
| AO | Additive outlier detection |
| LS | To detect level shifts in the time series |
| TC | To detect temporary changes |
| SLS | To consider seasonal level shifts while detecting outliers. |
| TSO | Basic time series outlier detection |
| CD | Cooks's distance for outlier detection using linear modelling |
| EV | Extreme value detection |

**Input:**

1. *inputValues* string = comma separated string of input values
2. *numInputs* numeric = type of input; Single input for = 0/1, Multiple inputs > 1
3. *alg* string = type of outlier detection; IO = Innovational outliers, AO = additive outliers, LS = level shift, TC = temporary changes, and SLS = seasonal level shifts.
4. *period(default=12)* numeric = type of time; yearly=1, monthly=12, weekly=52, daily=365
5. *interpolate* numeric = replacement parameter; 0 = default replacement using linear approximation, 1 = user defined interpolation for replacement, 2 = no replacement
6. *algGroup* numeric = algorithm group for interpolation; Simple = 0, Intermediate = 1, Advanced = 2
7. *algInterpolate* numeric = algorithm number in each group for interpolation; Group 0 = 0 to 5, Group 1 = 0,1, Group 2 = 0,1,2

8. *algSubType* numeric = to set the metric to be used as algorithm parameter for interpolation; Group 0 Alg 4/5 SubType = 0,1,2 Group 1 Alg 0/1 SubType = 0,1,2 Group 2 Alg 0 SubType = 0,1
9. *outputType* numeric = replacement values=0, positions of outlier values=1, replacement values and position of outlier values =2, replacement values and position of outlier values, where zeros in input are not treated as outliers = 3. The positions are identified as 1 for outlier in input, 0 otherwise.
10. *failSafe* numeric **(V1.3.0 only)** = choice of failsafe algorithm 0 = no fail safe, 1 = EV as fail-safe algorithm, 2 = run all algorithms until first valid output is produced.
11. *user* string = AIssisted username e.g., john@aissistedplanning.cloud.jedox.com
12. *apikey* string = AIssisted apikey

## Input V1.3.1, V1.3.2:

1. *InputValues* string = comma separated string of input values
2. *NumInputs* numeric = type of input; Single input for = 0/1, Multiple inputs > 1
3. *AlgorithmType* string = type of outlier detection; IO = Innovational outliers, AO = additive outliers, LS = level shift, TC = temporary changes, and SLS = seasonal level shifts.
4. *Period(default=12)* numeric = type of time; yearly=1, monthly=12, weekly=52, daily=365
5. *UseInterpolation* numeric = replacement parameter; 0 = default replacement using linear approximation, 1 = user defined interpolation for replacement, 2 = no replacement
6. *AlgGroup* numeric = algorithm group for interpolation; Simple = 0, Intermediate = 1, Advanced = 2
7. *AlgInterpolate* numeric = algorithm number in each group for interpolation; Group 0 = 0 to 5, Group 1 = 0,1, Group 2 = 0,1,2
8. *AlgSubType* numeric = to set the metric to be used as algorithm parameter for interpolation; Group 0 Alg 4/5 SubType = 0,1,2 Group 1 Alg 0/1 SubType = 0,1,2 Group 2 Alg 0 SubType = 0,1
9. *OutputType* numeric = replacement values=0, positions of outlier values=1, replacement values and position of outlier values =2, replacement values and position of outlier values, where zeros in input are not treated as outliers = 3. The positions are identified as 1 for outlier in input, 0 otherwise.
10. *FailSafe* numeric = choice of failsafe algorithm 0 = no fail safe, 1 = EV as fail-safe algorithm, 2 = run all algorithms until first valid output is produced.
11. *user* **(V1.3.1 only)** string = AIssisted username e.g., john@aissistedplanning.cloud.jedox.com
12. *apikey* **(V1.3.1 only)** string = AIssisted apikey
13. *AIssistedUser* **(V1.3.2 only)** string = AIssisted username e.g., john@aissistedplanning.cloud.jedox.com
14. *AIssistedKey* **(V1.3.2 only)** string = AIssisted apikey

**Output:** outputParameters.values[…] contains the following in the order as mentioned.

0. *Positions* = logical vector identifying outliers
1. *Values* = vector suggesting replacement values.

## Extrapolation Service:

**Name:** extrapolateJedox

**Aim:** To extrapolate time series/vectors in forward or backward directions.

**Use Case:** The extrapolation service is used for generating values in the future(forward) or past(backward) directions using different spline algorithms.

**Description:** The extrapolation service takes values and the type of algorithm as inputs. The user can decide which spline method to use; periodic, linear or cubic, as well as the direction of extrapolation.

**Call:**

https://${AIssisted_Planning_Server_IPorName}:12800/api/extrapolateJedox/v0.0.1

https://${AIssisted_Planning_Server_IPorName}:12800/api/extrapolateJedox/v1.0.0

**Input V0.0.1:**
1. *inputValues* string = comma separated string of input values
2. *numInputs* numeric = type of input; Single input = 0/1, Multiple inputs > 1
3. *alg* string = type of extrapolation method; per = periodic spline, lin = linear spline, cub = cubic spline.
4. *outputType* numeric = forward extrapolation (future values) =0, backward extrapolation (past values) =1, bidirectional extrapolation =2
5. *user* string = AIssisted username e.g., john@aissistedplanning.cloud.jedox.com
6. *apikey* string = AIssisted apikey

**Input V1.0.0:**
1. *InputValues* string = comma separated string of input values
2. *NumInputs* numeric = type of input; Single input = 0/1, Multiple inputs > 1
3. *AlgorithmType* string = type of extrapolation method; per = periodic spline, lin = linear spline, cub = cubic spline.
4. *OutputType* numeric = forward extrapolation (future values) =0, backward extrapolation (past values) =1, bidirectional extrapolation =2
5. *AIssistedUser* string = AIssisted username e.g., john@aissistedplanning.cloud.jedox.com
6. *AIssistedKey* string = AIssisted apikey

**Output:** outputParameters.values[…] contains the following in the order as mentioned.
1. *Values =* vector with extrapolated values.
2. *Positions =* vector with positions of extrapolated values.

## Predictive Forecasting (Time series Input=Values):

**Name:** predictiveForecast

**Aim:** To forecast values of given input(s) data.

**Use Case:** Forecasting is an essential part of planning sales and finances of any organization. This service provides the ability to use historic data and generate forecasts based on the past trends. The most common use of forecasting is budget and sales planning.

**Description:** The time series forecasting service uses several time series forecasting algorithms. These are simple linear modelling, seasonal naive forecasting, random walk with drift for forecasting, Holt Winters method, exponential smoothing and Autoregressive integrating moving averages (ARIMA).

The Linear modelling method fits a linear model and then makes prediction. Seasonal naive model is a basic time series model with consideration for seasonality of data. Holt winters method requires seasonal data with the number of inputs at least a multiple of period + 3 for quarterly data and period+5 for monthly data. Exponential smoothing, on the other hand, can also be used for non-seasonal data. The Random walk model uses Seasonal and Trend decomposition using Loess (STL) decomposition for time series analysis and forecast using random walk with drift (long-term trend assumed constant). This method requires a series length of >2*period (period - 12 monthly data, 4 quarterly data, 52 weekly data, 365 daily data). In case the choice of algorithm is not clear, automatic selection can also be performed. The service also calculates accuracy, root mean square error and lower/upper bounds for the forecasts.

**Call:**

https://${AIssisted_Planning_Server_IPorName_Port}/api/predictiveForecast/v1.2.0

https://${AIssisted_Planning_Server_IPorName_Port}/api/predictiveForecast/v1.3.0

https://${AIssisted_Planning_Server_IPorName_Port}/api/predictiveForecast/v1.3.1

**Input:**

1. *inputValues* string = comma separated string of input values

2. *numValues* numeric = number of values to be forecasted

3. *numPeriods* (OPTIONAL default = 0) numeric = number of periods

4. *alg* string = type of algorithm to be used; Linear Model (Time series; trend and season as inputs) = 1, Holt Winter = 2, Seasonal Naïve = 3, Exponential Smoothing = 4, ARIMA = 5, Random Walk with Drift = 6, Seasonal and Trend *decomposition* using Loess(STL) model = 7, generalized STL model = 8, neural network time series forecast = 9, TBATs model = 10, Croston's method = 11, Extrapolation = 12. Automatic Selection = 0/'all', best of classic (1,2,3,4,5,6,12) = "100"/"classic", best of advanced (7,8,9,10,11,12) = "101"/"innovative"/"novel".

5. *period* numeric = type of time; yearly=1, monthly=12, weekly=52, daily=365

6. *outputType* numeric = type of output; Forecast values, accuracy, root mean square error = 0, accuracy and algorithm used = 1, Forecast values with accuracy, lower bounds and upper bounds, normalized root mean square error = 2

7. *numInputs* numeric = type of input; Single input for forecasting = 0/1, Multiple inputs for forecasting > 1

8. *accThreshold* numeric = minimum accuracy for the forecast results

9**.** *interpolate* string = True/False for selecting whether missing values and zeros are to be replaced with meaningful values. If True, linear approximation is used for handling missing values. For handling trailing and leading zeros, last observation carry forward method is used.

10. *zeroRatio* numeric = Fraction of zeros and missing values allowed per time series.

11. *splitParameter* numeric = number of rows to be taken as test set for each time series.

12. *user* string = AIssisted username e.g., john@aissistedplanning.cloud.jedox.com

13. *apikey* string = AIssisted apikey

**Input V1.3.0, V1.3.1:**

1**.** *InputValues* string = comma separated string of input values

2. *NumberOfPredictions* numeric = number of values to be forecasted

3. *AlgorithmType* string = type of algorithm to be used; Linear Model (Time series; trend and season as inputs) = 1, Holt Winter = 2, Seasonal Naïve = 3, Exponential Smoothing = 4, ARIMA = 5, Random Walk with Drift = 6, Seasonal and Trend *decomposition* using Loess(STL) model = 7, generalized STL model = 8, neural network time series forecast = 9, TBATs model = 10, Croston's method = 11, Extrapolation = 12. Automatic Selection = 0/'all', best of classic (1,2,3,4,5,6,12) = "100"/"classic", best of advanced (7,8,9,10,11,12) = "101"/"innovative"/"novel".

4. *Period* numeric = type of time; yearly=1, monthly=12, weekly=52, daily=365

5. *OutputType* numeric = type of output; Forecast values, accuracy, root mean square error = 0, accuracy and algorithm used = 1, Forecast values with accuracy, lower bounds and upper bounds, normalized root mean square error = 2

6. *NumInputs* numeric = type of input; Single input for forecasting = 0/1, Multiple inputs for forecasting > 1

7. *AccurracyThreshold* numeric = minimum accuracy for the forecast results

8**.** *UseInterpolation* string = True/False for selecting whether missing values and zeros are to be replaced with meaningful values. If True, linear approximation is used for handling missing values. For handling trailing and leading zeros, last observation carry forward method is used.

9. *PercentageOfZeros* numeric = Fraction of zeros and missing values allowed per time series.

10. *NumberOfTestValues* numeric = number of rows to be taken as test set for each time series.

11. *user* **(V1.3.0 only)** string = AIssisted username e.g., john@aissistedplanning.cloud.jedox.com
12. *apikey* **(V1.3.0 only)** string = AIssisted apikey
13. *AIssistedUser* **(V1.3.1 only)** string = AIssisted username e.g., john@aissistedplanning.cloud.jedox.com
14. *AIssistedKey* **(V1.3.1 only)** string = AIssisted apikey

**Output:** outputParameters.values[…] contains the following in the order as mentioned.

0**.** *Forecast values* = forecasted values for one or more inputs using the user defined or automatically selected algorithm for accuracy greater than accThreshold, otherwise zero.

1. *Accuracy* = Accuracy on the training (or test set if applicable in case of version V1.0.1) set for the algorithm used. In case of best prediction for single input variable, accuracy of all algorithms is returned with the algorithm index of the highest accuracy. In case of best prediction for multiple inputs, highest accuracy along with the algorithm used for each input is returned.

2**.** *Lower bounds* = lower bound for forecasted values for one or more inputs using the user defined or automatically selected algorithm for accuracy greater than accThreshold, otherwise zero.

3. *Upper bounds* = Upper bound for forecasted values for one or more inputs using the user defined or automatically selected algorithm for accuracy greater than accThreshold, otherwise zero.

4. *Normalized Root Mean Square Error* = normalized root means square error on the training (or test set if applicable) set for the algorithm used.

# Jedox AIssisted ™ Planning- Python Web Services Documentation

## 1. Regression Service:

**Name:** AIssistedRegression

**Aim:** To train and/or save a Machine Learning regression model and predict target values.

**Use Case:** Regression service can be used when the predefined target values are known. The most common use case is driver-based planning. Given a set of drivers and target, the drivers are used as input for predicting the target. Another example is demand planning, such that using historic inventory data and related meta data as input, demand of products can be predicted.

**Description:** The regression service performs basic pre-processing (if required) on input data and trains a regression model for predicting target values. This AIssisted service returns the predicted values, training and test evaluation metrics corresponding to the requested output type. The service accepts mixed data type; categorical, decimal/float type numbers and integer/whole numbers and performs (any) necessary transformations.

The regression services only perform basic feature selection in the form of correlation analysis and variance check. The features that are highly correlated to each other are dropped except for one. Similarly, features with the least variance are also dropped. These thresholds for these to filtering criteria are taken as user input, explained in detail later.

There are seven regression algorithms that can be used: Linear regression, Random Forest, Support Vector Regression, ExtraTrees, AdaBoost, Gradient Boosting and KNN regression. The service also allows for automatic selection of regression model based on training accuracy, i.e., the algorithm that gives highest accuracy on training data is used for making predictions. The accuracy for the regression predictions is calculated as the percentage of hits. A predicted value is considered as a hit if it lies within the error margin, which is taken as user input.

**Call:**
https://${AIssisted_Planning_Server_IPorName}:12800/api/AIssistedRegression/${version}

| ALGORITHM | WEB SERVICE INPUT |
|---|---|
| Best Selection | 0 |
| Linear Regression | 1 |
| Support Vector Machine | 2 |
| Random Forest | 3 |
| Extra Trees | 4 |
| AdaBoost Regressor | 5 |
| Gradient Boosting Regressor | 6 |
| KNN Regression | 7 |

TABLE 5: Algorithms used in Regression Web Service

| WEB SERVICE NAME | SERVICE INPUTS | DESCRIPTION |
|---|---|---|
| AIssistedRegression | feature_names, train_fatures, test_features, target, alg, correlation_threshold, variance_threshold, cv, use_bagging, optimize_paramters, normalize, model_name, result_file_name, class_weight, fix_random_state, val_size, cat_input, user, api_key | Data Input: comma separated strings Data output: Predictions and evaluation metrics, output file, saved model (based on user selection) Trained Model: can be saved based on user selection |
| | | |
| | | |
| | | |

TABLE 6: Description of Regression Web Services

**Input:**

1. feature_names (*string*) = comma separated string of feature names.
2. file_name (*string*) = path and name of input file
3. train_features (*string*) = comma separated string of training features. Features are supposed to be concatenated column wise.
4. test_features (*string*) = comma separated string of test features.
5. target (*string*) = name of the target feature (to be predicted).
6. alg (*numeric*) = regression algorithm to be used. Refer Table 3.
7. correlation_threshold (*float*) = correlation coefficient, between 0 and 1. The features with correlation greater or equal to this input are dropped.
8. variance_threshold (*string*) = features variance, between 0 and 1. The features with variance lesser or equal to this input are dropped.
9. cv (*numeric*) = cross validation folds used when using optimize_parameters is True.
10. use_bagging (*string*) = True or False. When set to "True" bagging is used.
11. optimize_parameters (*integer*) = Number of optimization iterations to run. Random search for optimizing regression model. Requires cv>1.
12. normalize (*string*) = True or False. When set to "True", data is normalized before training.
13. model_name (*string*) = OPTIONAL, DEFAULT = "" name of the model file
14. result_file_name (*string*) = OPTIONAL, DEFAULT = "" name of the result file
15. fix_random_state (*string*) = To use a seed (42) for all randomized operations within web service or not.
16. val_size (float)= Percentage of training set to use for validation.
17. cat_input (string)= Comma separated string of features to be 1-hot encoded.
18. user (*string*)= AIssisted username e.g., john@aissistedplanning.cloud.jedox.com
19. api_key (*string*)= AIssisted apikey

**Output:** outputParameters.values[ ] contains the following in the order as mentioned

1. success = True or False. Status of the code execution
2. flag = 0, 1 or 2. 0 = no error, 1 = handled error, 2 = unhandled error
3. Train Scores = evaluation metrics for training, in the following order
    a. accuracy, for each method when best selection
    b. mae (mean absolute error) of the method used
    c. mse (mean squared error) of the method used
    d. maxe (maximum error) of the method used
    e. r2 (R2 score) of the method used
    f. rmse (root mean squared error) for the method used
    g. nrmse (normalized root mean squared error) for the method used
4. Time Stamp = date and time
5. Test Indices = index of the rows in test data/features
6. Test Predictions = predicted values
7. Test Scores = evaluation metrics for test data when target is given for testing, otherwise all the metrics returned are zero. The order is same as training scores (point 3)
8. Algorithm = regression algorithm used
9. Model Name = when corresponding input is given, name of the stored model otherwise "No model stored"
10. Result File = when corresponding input is given, name of the result file otherwise "No result file written"

11. Validation Accuracies = The accuracies of the chosen algorithm on the training set.
12. Validation Scores = Same metrics for training set but for validation set.
13. Validation Predictions = Best model predictions for validation set.
14. Dropped Features = Features that were dropped for failing to meet the correlation or variance threshold set.