Package 'prophet'

November 8, 2017

Title Automatic Forecasting Procedure

Version 0.2.1

Date 2017-11-08

Description Implements a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly and weekly seasonality, plus holidays. It works best with daily periodicity data with at least one year of historical data. Prophet is robust to missing data, shifts in the trend, and large outliers.

Depends R (>= 3.2.3), Rcpp (>= 0.12.0)

Imports dplyr (>= 0.5.0), extraDistr, ggplot2, grid, rstan (>= 2.14.0), scales, stats, tidyr (>= 0.6.1)

Suggests knitr, testthat, readr

License BSD_3_clause + file LICENSE

LazyData true

RoxygenNote 5.0.1

VignetteBuilder knitr

SystemRequirements C++11

NeedsCompilation yes

Author Sean Taylor [cre, aut], Ben Letham [aut]

Maintainer Sean Taylor <sjt@fb.com>

Repository CRAN

Date/Publication 2017-11-08 22:39:24 UTC

R topics documented:

`add_regressor` *Add an additional regressor to be used for fitting and predicting.*

Description

The dataframe passed to 'fit' and 'predict' will have a column with the specified name to be used as a regressor. When standardize='auto', the regressor will be standardized unless it is binary. The regression coefficient is given a prior with the specified scale parameter. Decreasing the prior scale will add additional regularization. If no prior scale is provided, holidays.prior.scale will be used.

Usage

```
add_regressor(m, name, prior.scale = NULL, standardize = "auto")
```

Arguments

`m`	Prophet object.
`name`	String name of the regressor
`prior.scale`	Float scale for the normal prior. If not provided, holidays.prior.scale will be used.
`standardize`	Bool, specify whether this regressor will be standardized prior to fitting. Can be 'auto' (standardize if not binary), True, or False.

Value

The prophet model with the regressor added.

add_seasonality *Add a seasonal component with specified period, number of Fourier components, and prior scale.*

Description

Increasing the number of Fourier components allows the seasonality to change more quickly (at risk of overfitting). Default values for yearly and weekly seasonalities are 10 and 3 respectively.

Usage

```
add_seasonality(m, name, period, fourier.order, prior.scale = NULL)
```

Arguments

m	Prophet object.
name	String name of the seasonality component.
period	Float number of days in one period.
fourier.order	Int number of Fourier components to use.
prior.scale	Float prior scale for this component.

Details

Increasing prior scale will allow this seasonality component more flexibility, decreasing will dampen it. If not provided, will use the seasonality.prior.scale provided on Prophet initialization (defaults to 10).

Value

The prophet model with the seasonality added.

cross_validation *Cross-validation for time series.*

Description

Computes forecasts from historical cutoff points. Beginning from initial, makes cutoffs with a spacing of period up to (end - horizon).

Usage

```
cross_validation(model, horizon, units, period = NULL, initial = NULL)
```

Arguments

model	Fitted Prophet model.
horizon	Integer size of the horizon
units	String unit of the horizon, e.g., "days", "secs".
period	Integer amount of time between cutoff dates. Same units as horizon. If not provided, 0.5 * horizon is used.
initial	Integer size of the first training period. If not provided, 3 * horizon is used. Same units as horizon.

Details

When period is equal to the time interval of the data, this is the technique described in https://robjhyndman.com/hyndsight/tscv
.

Value

A dataframe with the forecast, actual value, and cutoff date.

---

fit.prophet *Fit the prophet model.*

Description

This sets m$params to contain the fitted model parameters. It is a list with the following elements: k (M array): M posterior samples of the initial slope. m (M array): The initial intercept. delta (MxN matrix): The slope change at each of N changepoints. beta (MxK matrix): Coefficients for K seasonality features. sigma_obs (M array): Noise level. Note that M=1 if MAP estimation.

Usage

```
fit.prophet(m, df, ...)
```

Arguments

m	Prophet object.
df	Data frame.
...	Additional arguments passed to the optimizing or sampling functions in Stan.

make_future_dataframe *Make dataframe with future dates for forecasting.*

Description

Make dataframe with future dates for forecasting.

Usage

```
make_future_dataframe(m, periods, freq = "day", include_history = TRUE)
```

Arguments

m	Prophet model object.
periods	Int number of periods to forecast forward.
freq	'day', 'week', 'month', 'quarter', 'year', 1(1 sec), 60(1 minute) or 3600(1 hour).
include_history	Boolean to include the historical dates in the data frame for predictions.

Value

Dataframe that extends forward from the end of m$history for the requested number of periods.

plot.prophet *Plot the prophet forecast.*

Description

Plot the prophet forecast.

Usage

```
## S3 method for class 'prophet'
plot(x, fcst, uncertainty = TRUE, plot_cap = TRUE,
  xlabel = "ds", ylabel = "y", ...)
```

Arguments

x	Prophet object.
fcst	Data frame returned by predict(m, df).
uncertainty	Boolean indicating if the uncertainty interval for yhat should be plotted. Must be present in fcst as yhat_lower and yhat_upper.
plot_cap	Boolean indicating if the capacity should be shown in the figure, if available.
xlabel	Optional label for x-axis
ylabel	Optional label for y-axis
...	additional arguments

Value

A ggplot2 plot.

Examples

```
## Not run:
history <- data.frame(ds = seq(as.Date('2015-01-01'), as.Date('2016-01-01'), by = 'd'),
                      y = sin(1:366/200) + rnorm(366)/10)
m <- prophet(history)
future <- make_future_dataframe(m, periods = 365)
forecast <- predict(m, future)
plot(m, forecast)

## End(Not run)
```

---

plot_forecast_component

Plot a particular component of the forecast.

Description

Plot a particular component of the forecast.

Usage

```
plot_forecast_component(fcst, name, uncertainty = TRUE, plot_cap = FALSE)
```

Arguments

fcst	Dataframe output of 'predict'.
name	String name of the component to plot (column of fcst).
uncertainty	Boolean to plot uncertainty intervals.
plot_cap	Boolean indicating if the capacity should be shown in the figure, if available.

Value

A ggplot2 plot.

predict.prophet *Predict using the prophet model.*

Description

Predict using the prophet model.

Usage

```
## S3 method for class 'prophet'
predict(object, df = NULL, ...)
```

Arguments

object	Prophet object.
df	Dataframe with dates for predictions (column ds), and capacity (column cap) if logistic growth. If not provided, predictions are made on the history.
...	additional arguments.

Value

A dataframe with the forecast components.

Examples

```
## Not run:
history <- data.frame(ds = seq(as.Date('2015-01-01'), as.Date('2016-01-01'), by = 'd'),
                      y = sin(1:366/200) + rnorm(366)/10)
m <- prophet(history)
future <- make_future_dataframe(m, periods = 365)
forecast <- predict(m, future)
plot(m, forecast)

## End(Not run)
```

predictive_samples *Sample from the posterior predictive distribution.*

Description

Sample from the posterior predictive distribution.

Usage

```
predictive_samples(m, df)
```

Arguments

m	Prophet object.
df	Dataframe with dates for predictions (column ds), and capacity (column cap) if logistic growth.

Value

A list with items "trend", "seasonal", and "yhat" containing posterior predictive samples for that component. "seasonal" is the sum of seasonalities, holidays, and added regressors.

---

prophet *Prophet forecaster.*

Description

Prophet forecaster.

Usage

```
prophet(df = NULL, growth = "linear", changepoints = NULL,
  n.changepoints = 25, yearly.seasonality = "auto",
  weekly.seasonality = "auto", daily.seasonality = "auto",
  holidays = NULL, seasonality.prior.scale = 10,
  holidays.prior.scale = 10, changepoint.prior.scale = 0.05,
  mcmc.samples = 0, interval.width = 0.8, uncertainty.samples = 1000,
  fit = TRUE, ...)
```

Arguments

df	(optional) Dataframe containing the history. Must have columns ds (date type) and y, the time series. If growth is logistic, then df must also have a column cap that specifies the capacity at each ds. If not provided, then the model object will be instantiated but not fit; use fit.prophet(m, df) to fit the model.
growth	String 'linear' or 'logistic' to specify a linear or logistic trend.
changepoints	Vector of dates at which to include potential changepoints. If not specified, potential changepoints are selected automatically.
n.changepoints	Number of potential changepoints to include. Not used if input 'changepoints' is supplied. If 'changepoints' is not supplied, then n.changepoints potential changepoints are selected uniformly from the first 80 percent of df$ds.
yearly.seasonality	Fit yearly seasonality. Can be 'auto', TRUE, FALSE, or a number of Fourier terms to generate.
weekly.seasonality	Fit weekly seasonality. Can be 'auto', TRUE, FALSE, or a number of Fourier terms to generate.

`daily.seasonality`
: Fit daily seasonality. Can be 'auto', TRUE, FALSE, or a number of Fourier terms to generate.

`holidays`
: data frame with columns holiday (character) and ds (date type)and optionally columns lower_window and upper_window which specify a range of days around the date to be included as holidays. lower_window=-2 will include 2 days prior to the date as holidays. Also optionally can have a column prior_scale specifying the prior scale for each holiday.

`seasonality.prior.scale`
: Parameter modulating the strength of the seasonality model. Larger values allow the model to fit larger seasonal fluctuations, smaller values dampen the seasonality. Can be specified for individual seasonalities using add_seasonality.

`holidays.prior.scale`
: Parameter modulating the strength of the holiday components model, unless overridden in the holidays input.

`changepoint.prior.scale`
: Parameter modulating the flexibility of the automatic changepoint selection. Large values will allow many changepoints, small values will allow few changepoints.

`mcmc.samples`
: Integer, if greater than 0, will do full Bayesian inference with the specified number of MCMC samples. If 0, will do MAP estimation.

`interval.width`
: Numeric, width of the uncertainty intervals provided for the forecast. If mcmc.samples=0, this will be only the uncertainty in the trend using the MAP estimate of the extrapolated generative model. If mcmc.samples>0, this will be integrated over all model parameters, which will include uncertainty in seasonality.

`uncertainty.samples`
: Number of simulated draws used to estimate uncertainty intervals.

`fit`
: Boolean, if FALSE the model is initialized but not fit.

`...`
: Additional arguments, passed to `fit.prophet`

Value

A prophet model.

Examples

```
## Not run:
history <- data.frame(ds = seq(as.Date('2015-01-01'), as.Date('2016-01-01'), by = 'd'),
                      y = sin(1:366/200) + rnorm(366)/10)
m <- prophet(history)

## End(Not run)
```

prophet_plot_components *Plot the components of a prophet forecast. Prints a ggplot2 with panels for trend, weekly and yearly seasonalities if present, and holidays if present.*

Description

Plot the components of a prophet forecast. Prints a ggplot2 with panels for trend, weekly and yearly seasonalities if present, and holidays if present.

Usage

```
prophet_plot_components(m, fcst, uncertainty = TRUE, plot_cap = TRUE,
  weekly_start = 0, yearly_start = 0)
```

Arguments

m	Prophet object.
fcst	Data frame returned by predict(m, df).
uncertainty	Boolean indicating if the uncertainty interval should be plotted for the trend, from fcst columns trend_lower and trend_upper.
plot_cap	Boolean indicating if the capacity should be shown in the figure, if available.
weekly_start	Integer specifying the start day of the weekly seasonality plot. 0 (default) starts the week on Sunday. 1 shifts by 1 day to Monday, and so on.
yearly_start	Integer specifying the start day of the yearly seasonality plot. 0 (default) starts the year on Jan 1. 1 shifts by 1 day to Jan 2, and so on.

Value

Invisibly return a list containing the plotted ggplot objects

simulated_historical_forecasts *Simulated historical forecasts.*

Description

Make forecasts from k historical cutoff points, working backwards from (end - horizon) with a spacing of period between each cutoff.

Usage

```
simulated_historical_forecasts(model, horizon, units, k, period = NULL)
```

Arguments

model	Fitted Prophet model.
horizon	Integer size of the horizon
units	String unit of the horizon, e.g., "days", "secs".
k	integer number of forecast points
period	Integer amount of time between cutoff dates. Same units as horizon. If not provided, will use 0.5 * horizon.

Value

A dataframe with the forecast, actual value, and cutoff date.

Index