SKFORECAST: PREDICCIÓN DE SERIES TEMPORALES CON MODELOS SCIKIT-LEARN

Joaquín Amat Rodrigo
Javier Escobar Ortiz

Skforecast: GitHub · Skforecast: Docs · www.cienciadedatos.net

BEHIND THE SCENES

Javier Escobar Ortiz

Data Scientist @IKEA 🇸🇪🌭

LinkedIn

javier.escobar.ortiz@gmail.com

Joaquín Amat Rodrigo

Data Scientist @Veeva Systems

LinkedIn

j.amatrodrigo@gmail.com

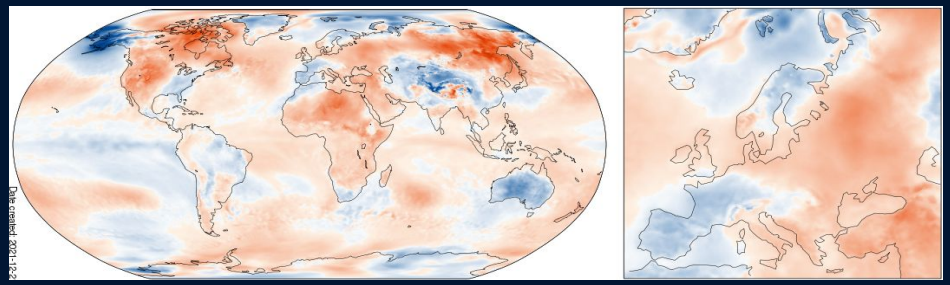
Skforecast: GitHub Skforecast: Docs www.cienciadedatos.net

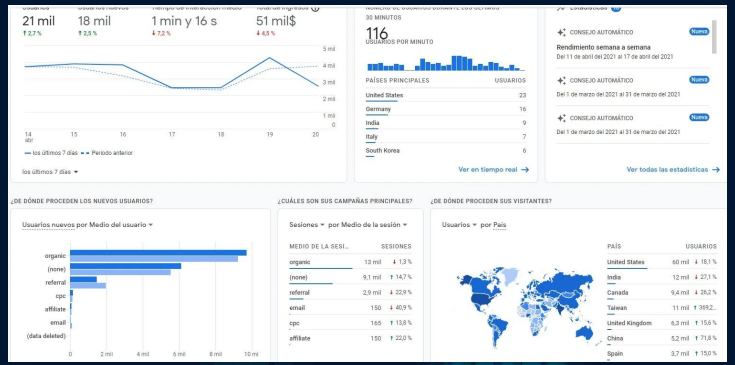
ÍNDICE

- Series temporales
- Forecasting
- Horizonte de predicción: *single-step* y *multi-step*
    - Modelos *multi-step* recursivos
    - Modelos *multi-step* directos
- Validación de modelos de forecasting
- Búsqueda de hiperparámetros
- Intervalos de predicción
- Modelos multiseries y multivariante
- Otras funcionalidades de Skforecast
- Material adicional

SERIES TEMPORALES

Una serie temporal (*time series*) es una sucesión de datos ordenados cronológicamente y espaciados a intervalos iguales o desiguales.

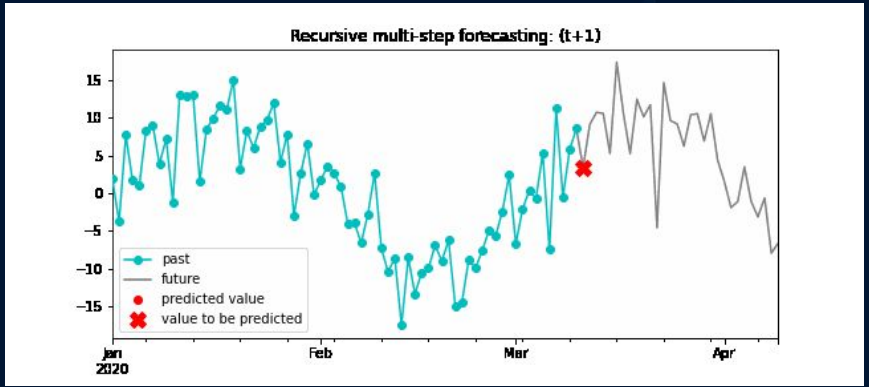




[Skforecast: GitHub](#) [Skforecast: Docs](#) [www.cienciadedatos.net](#)

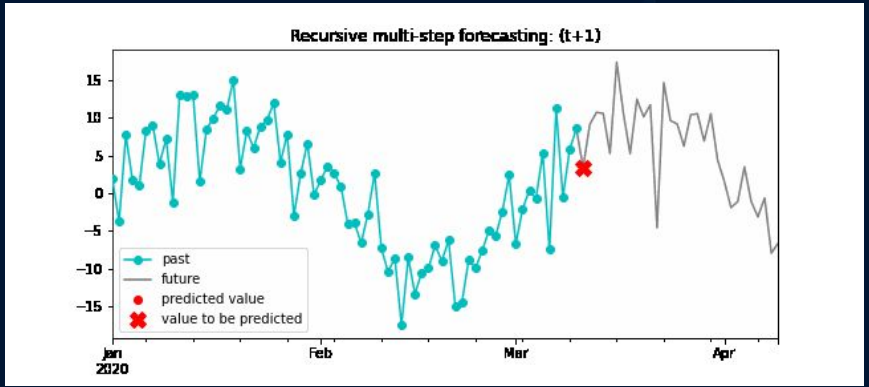
FORECASTING

El proceso de *forecasting* consiste en predecir el valor futuro de una serie temporal, bien modelando la serie únicamente en función de su comportamiento pasado (autorregresivo) o empleando otras variables externas.



FORECASTING

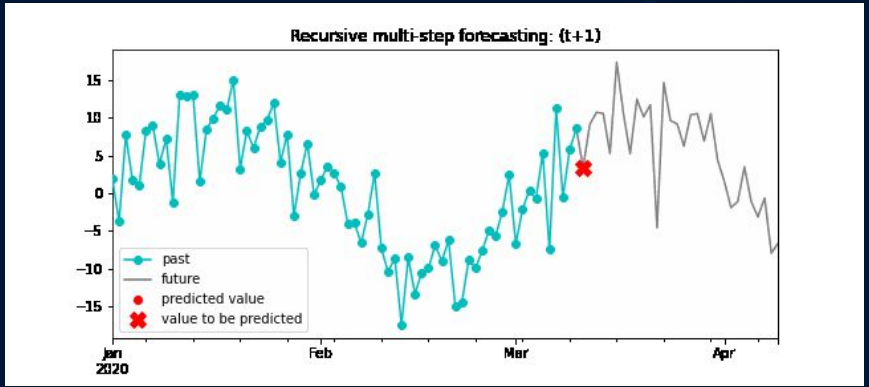
El proceso de *forecasting* consiste en predecir el valor futuro de una serie temporal, bien modelando la serie únicamente en función de su comportamiento pasado (autorregresivo) o empleando otras variables externas.



Para crear un modelo de forecasting, se utilizan datos históricos con el objetivo de obtener una representación matemática capaz de predecir futuros valores. Esta idea se fundamenta en la asunción de que el comportamiento futuro de un fenómeno se puede explicar a partir de su comportamiento pasado.

FORECASTING

El proceso de *forecasting* consiste en predecir el valor futuro de una serie temporal, bien modelando la serie únicamente en función de su comportamiento pasado (autorregresivo) o empleando otras variables externas.



Para crear un modelo de forecasting, se utilizan datos históricos con el objetivo de obtener una representación matemática capaz de predecir futuros valores. Esta idea se fundamenta en la asunción de que el comportamiento futuro de un fenómeno se puede explicar a partir de su comportamiento pasado.

Esto raramente ocurre en la realidad o, al menos, no en su totalidad.

ESTRATEGIAS DE FORECASTING

Modelos Estadísticos-Econométricos

Autoregression (AR)
Moving Average (MA)
Autoregressive Moving Average (ARMA)
Autoregressive Integrated Moving Average (ARIMA)
Simple Exponential Smoothing (SES)
Holt Winter's Exponential Smoothing (HWES)

Modelos de Machine learning

Lasso
Ridge
Random Forest
Gradient Boosting
SVM
Neural networks
LSTM

Modelos de step único
Modelos multi-step recursivo
Modelos multi-step directo

ESTRATEGIAS DE FORECASTING

Modelos Estadísticos-Econométricos

Autoregression (AR)
Moving Average (MA)
Autoregressive Moving Average (ARMA)
Autoregressive Integrated Moving Average (ARIMA)
Simple Exponential Smoothing (SES)
Holt Winter's Exponential Smoothing (HWES)

Modelos de Machine learning

Lasso
Ridge
Random Forest
Gradient Boosting
SVM
Neural networks
LSTM

Modelos de step único
Modelos multi-step recursivo
Modelos multi-step directo

PREDICCIÓN

Single step

El objetivo es predecir únicamente el siguiente valor de la serie, **t+1**.

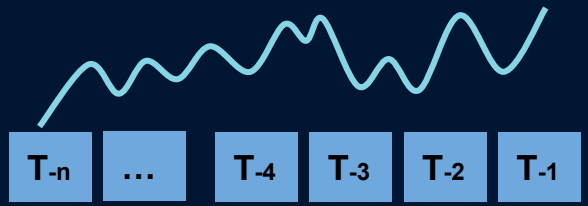
Multi step

El objetivo es predecir los ***n*** siguientes valores de la serie.

- Estrategia multi-step recursiva
- Estrategia multi-step directa

PREDICCIÓN SINGLE-STEP

El objetivo es predecir únicamente el siguiente valor de la serie.



Valor observado

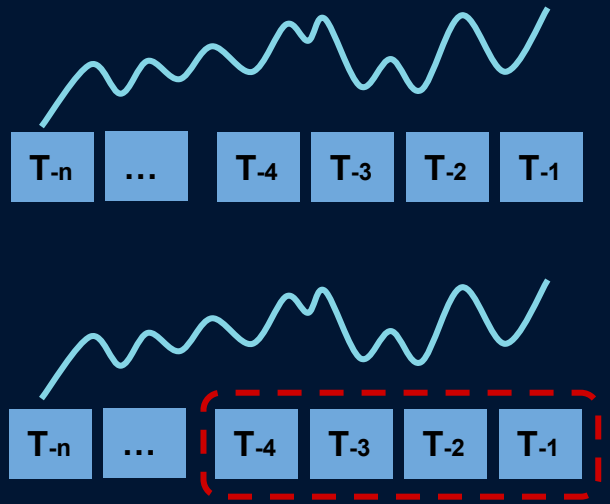
Predictores (lags)

Modelo para step +1

Valor predicho

PREDICCIÓN SINGLE-STEP

El objetivo es predecir únicamente el siguiente valor de la serie.



Valor observado

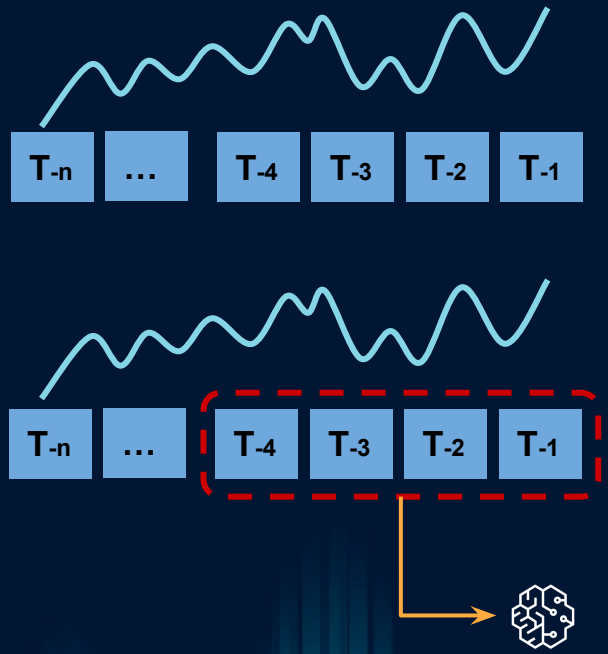
Predictores (lags)

Modelo para step +1

Valor predicho

PREDICCIÓN SINGLE-STEP

El objetivo es predecir únicamente el siguiente valor de la serie.



Valor observado

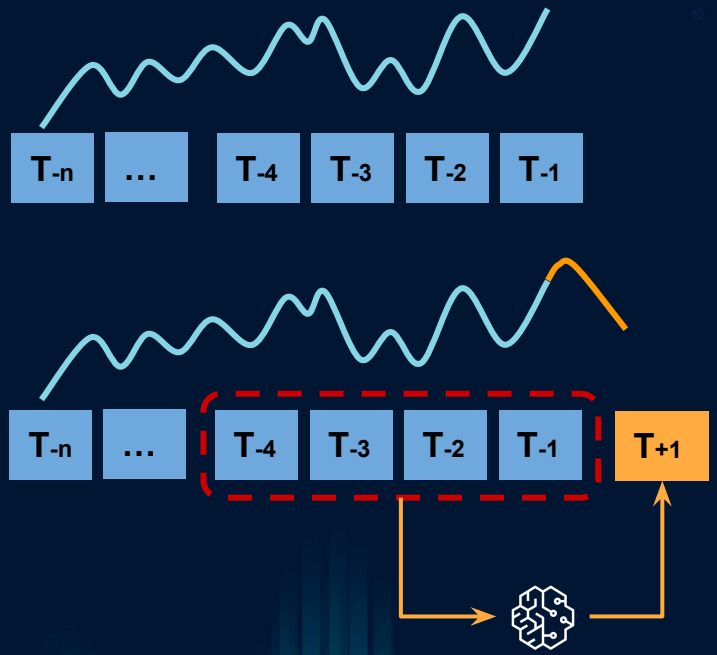
Predictores (lags)

Modelo para step +1

Valor predicho

PREDICCIÓN SINGLE-STEP

El objetivo es predecir únicamente el siguiente valor de la serie.



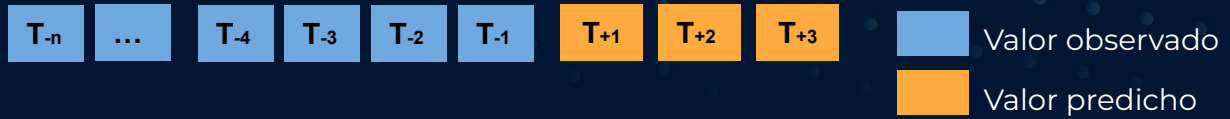
Valor observado

Predictores (lags)

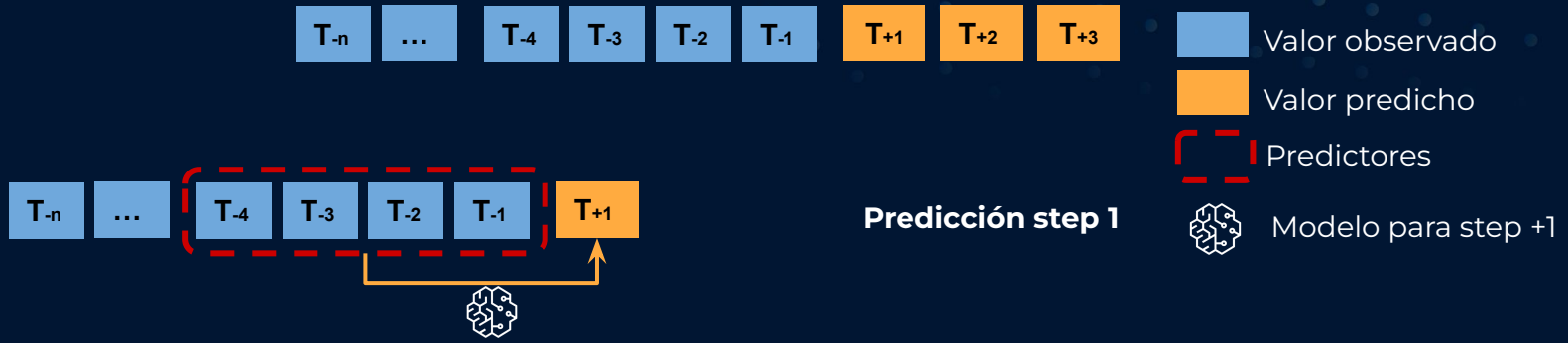
Modelo para step +1

Valor predicho

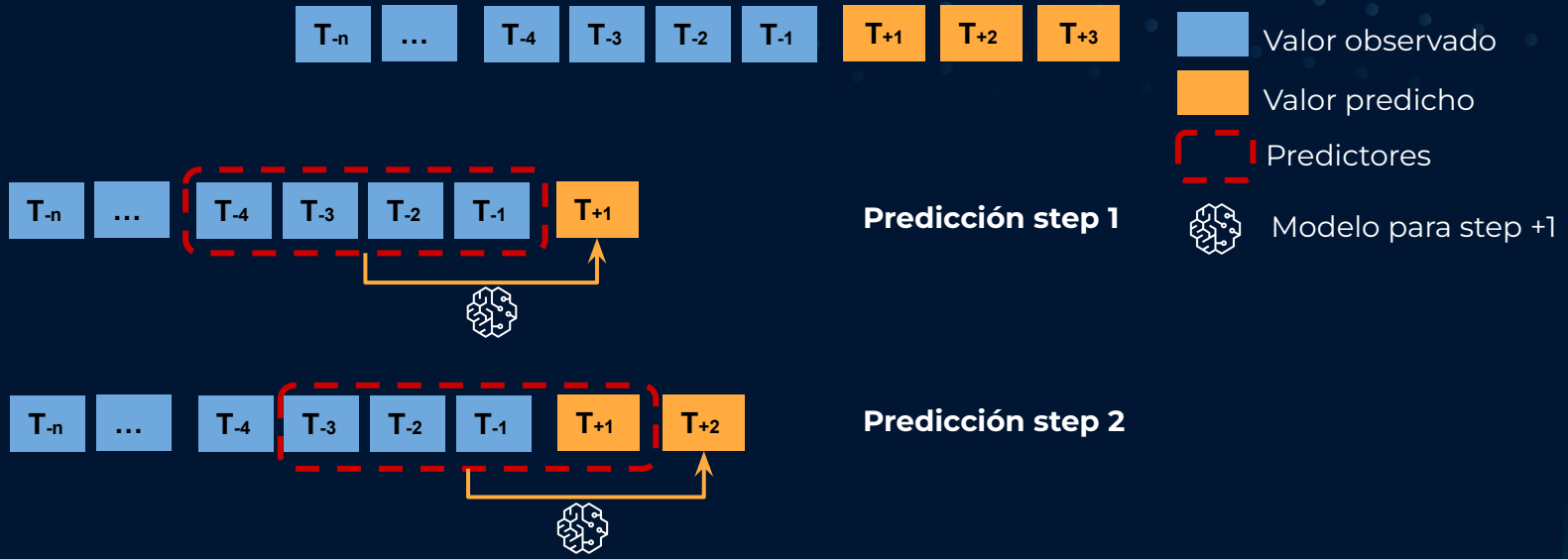
PREDICCIÓN MULTI-STEP RECURSIVA



PREDICCIÓN MULTI-STEP RECURSIVA



PREDICCIÓN MULTI-STEP RECURSIVA



Skforecast: GitHub Skforecast: Docs www.cienciadedatos.net

PREDICCIÓN MULTI-STEP RECURSIVA

Skforecast: GitHub Skforecast: Docs www.cienciadedatos.net

ForecasterAutoreg

```
# Crear y ajustar forecaster
# ==============================================================================
forecaster = ForecasterAutoreg(
                regressor        = RandomForestRegressor(random_state=123),
                lags             = 12,
                transformer_y    = None,    # transformación serie y
                transformer_exog = None,    # transformación exógenas
                weight_func      = None     # pesos observaciones serie temporal según índice
             )

forecaster.fit(y=data_train['y'], exog=None)

# Predicciones
# ==============================================================================
steps = 36 # steps a predecir
predicciones = forecaster.predict(steps=steps, last_window=None)
```

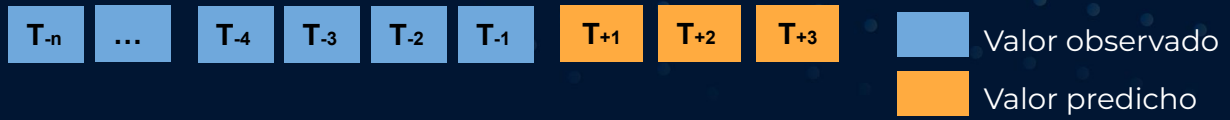
ForecasterAutoregCustom

```
# Función custom para crear los predictores (fila a fila)
# ==============================================================================
def create_predictors(y): # serie y como argumento
    """
    Crear los 10 primeros lags de una serie temporal.
    Calcular la media móvil con 20 observaciones.
    """
    lags = y[-1:-11:-1]
    mean = np.mean(y[-20:])
    predictors = np.hstack([lags, mean])

    return predictors
```

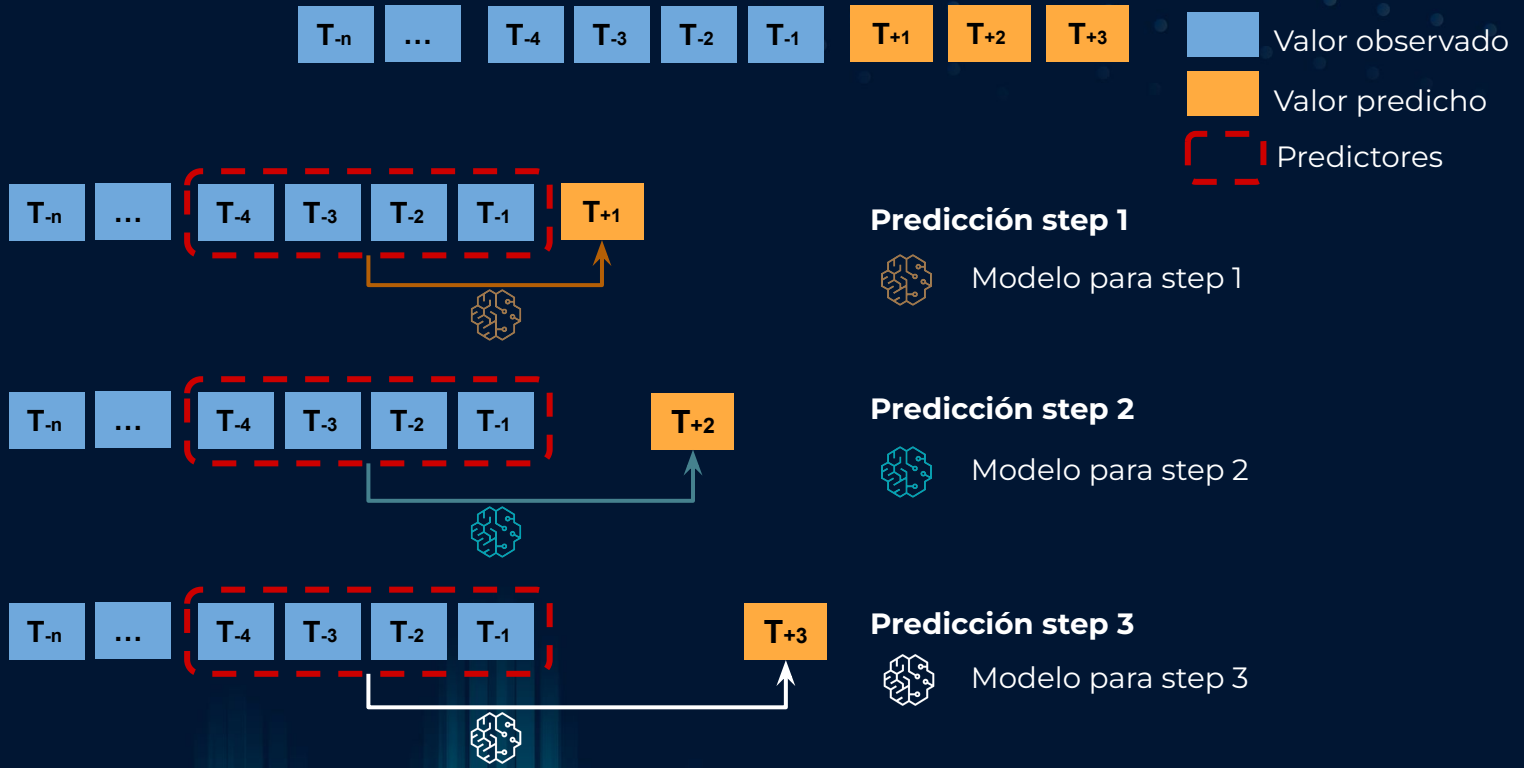
ForecasterAutoregCustom

```
# Crear y ajustar forecaster
# ==============================================================================
forecaster = ForecasterAutoregCustom(
                regressor        = LGBMRegressor(random_state=123),
                fun_predictors   = create_predictors, # función para crear predictores
                window_size      = 20,                # ventana observaciones necesita la función
                transformer_y    = None,
                transformer_exog = None,
                weight_func      = None
             )
forecaster.fit(y=data_train['y'], exog=None)

# Predicciones
# ==============================================================================
steps = 36 # steps a predecir, 36 meses
predicciones = forecaster.predict(steps=steps, last_window=None)
```

PREDICCIÓN MULTI-STEP DIRECTA



PREDICCIÓN MULTI-STEP DIRECTA



Skforecast: GitHub Skforecast: Docs www.cienciadedatos.net

ForecasterAutoregDirect

```
# Crear y ajustar Forecaster
# ==============================================================================
forecaster = ForecasterAutoregDirect(
                regressor        = Ridge(random_state=123),
                steps            = 36,    # cantidad de regresores, 1 regresor por step
                lags             = 15,
                transformer_y    = None,
                transformer_exog = None,
                weight_func      = None
             )
forecaster.fit(y=data_train['y'], exog=None)

# Predicciones
# ==============================================================================
steps = 36 # steps a predecir, 36 meses
predicciones = forecaster.predict(steps=steps, last_window=None)
```

COMPARATIVA PREDICCIÓN MULTI-STEP

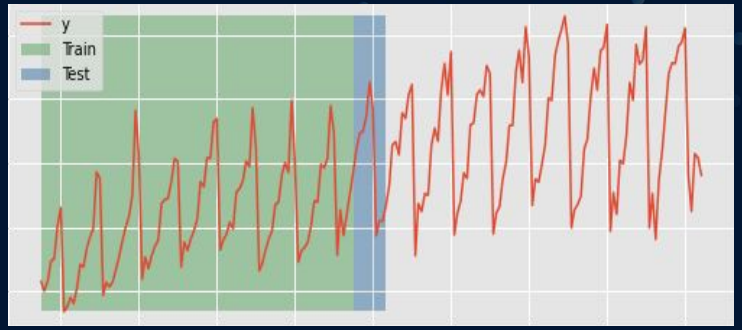
- Ninguno de los métodos de predicción supera a otro en todos los escenarios. Depende del caso de uso.

- El método multi-step directo requiere entrenar un modelo por cada step, por lo que tiene mayores requerimientos computacionales.

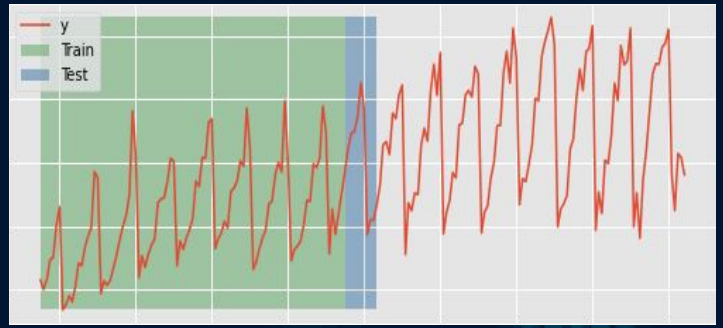
- Con el método multi-step directo se tiene que definir de antemano el horizonte de predicción, cosa que no es necesaria con el método multi-step recursivo.

VALIDACIÓN DE MODELOS (BACKTESTING)

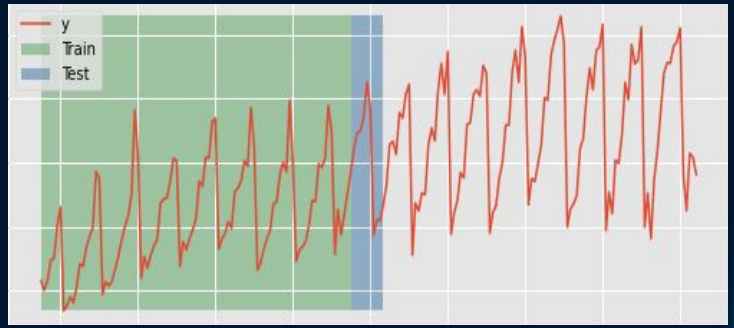
Backtesting sin reentrenamiento



Backtesting con reentrenamiento (time series cross validation, rolling origen)



Backtesting con reentrenamiento (fixed origen)



Backtesting

```
# Backtesting forecaster
# ==============================================================================
metric, preds_backtest = backtesting_forecaster(
                             forecaster         = forecaster,            # forecaster
                             y                  = data['y'],             # serie completa
                             exog               = None,                  # variables exógenas
                             steps              = 12,                    # nº steps a predecir
                             metric             = 'mean_absolute_error', # métrica
                             initial_train_size = len(data_train),       # observaciones para
entrenamiento inicial
                             fixed_train_size   = True,    # fix/moving train size
                             refit              = True,    # reentrenar después de cada predicción
                             verbose            = True     # verbose
                         )
```

Grid search

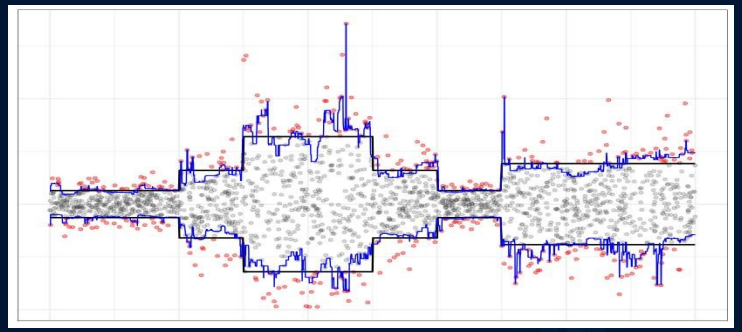
```
# Grid search de hiperparámetros
# ==============================================================================
forecaster = ForecasterAutoreg(
                 regressor       = RandomForestRegressor(random_state=123),
                 lags            = 12, # Este valor será reemplazado en el grid search
                 transformer_y   = StandardScaler(),
                 transformer_exog = None,
                 weight_func     = None
             )

# Lags utilizados como predictores
lags_grid = [6, 12, 18]

# Hiperparámetros del regresor
param_grid = {'n_estimators': [100, 200],
              'max_depth': [3, 5, 10]}
```

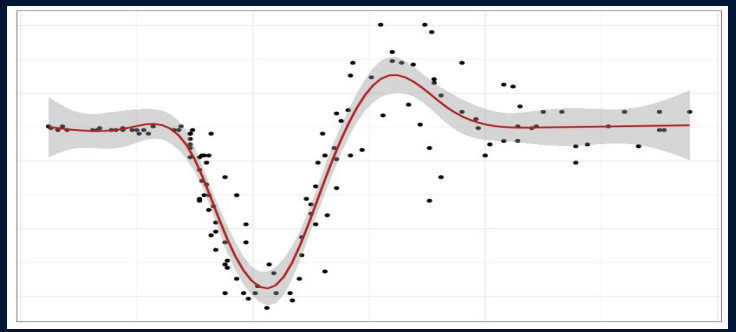
Grid search

```
# Grid search de hiperparámetros
# ==============================================================================
results_grid = grid_search_forecaster(
                   forecaster         = forecaster,          # forecaster
                   y                  = data['y'],           # serie completa
                   exog               = None,                # variables exógenas
                   param_grid         = param_grid,          # grid hiperparámetros
                   lags_grid          = lags_grid,           # grid lags
                   steps              = 12,                  # nº steps a predecir
                   metric             = mean_absolute_error, # Callable métrica
                   initial_train_size = len(data_train),     # observaciones para train inicial
                   fixed_train_size   = True,                # fix/moving train size
                   refit              = True,     # reentrenar después de cada predicción
                   return_best        = True,     # reentrenar con mejor combinación el forecaster
                   verbose            = False     # Verbose False
               )
```

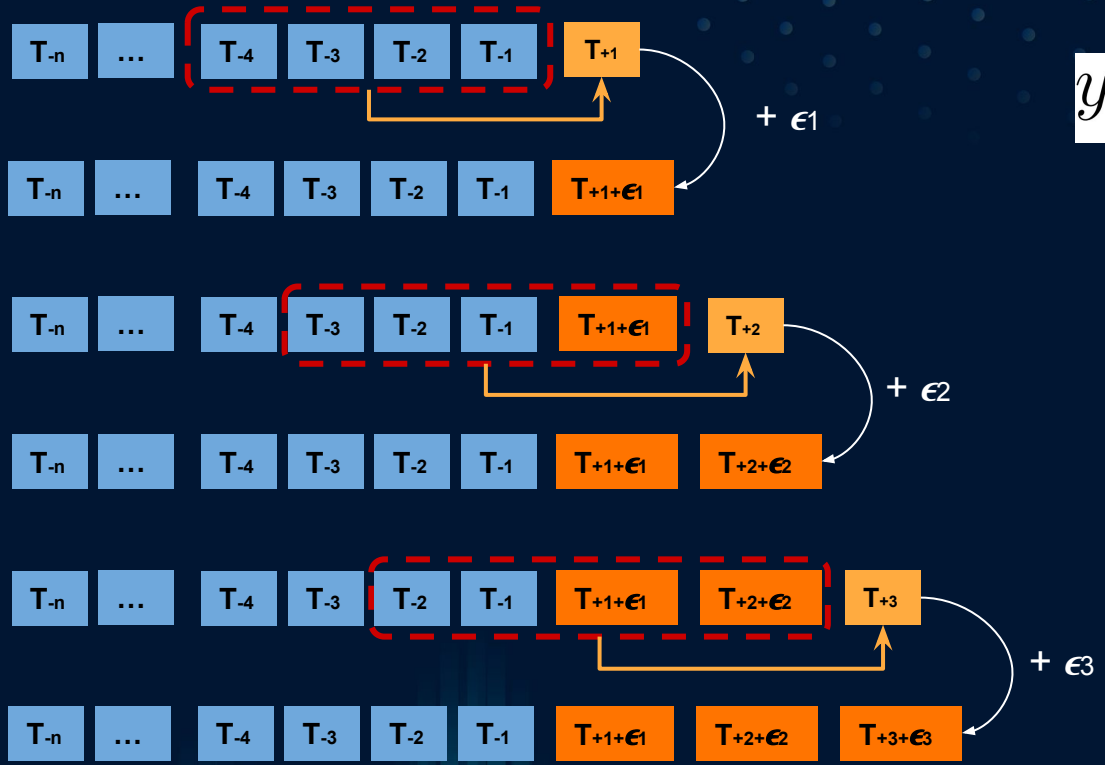
FORECASTING PROBABILíSTICO: INTERVALOS DE PREDICCIÓN

Un intervalo de predicción define el intervalo dentro del cual es de esperar que se encuentre el verdadero valor de y con una determinada probabilidad. Por ejemplo, es de esperar que el intervalo de predicción del 98% contenga el verdadero valor de la predicción con un 98% de probabilidad.





INTERVALOS DE PREDICCIÓN: BOOTSTRAPPED RESIDUALS



$$y_t = \hat{y}_{t|t-1} + \epsilon_1$$

Valor observado

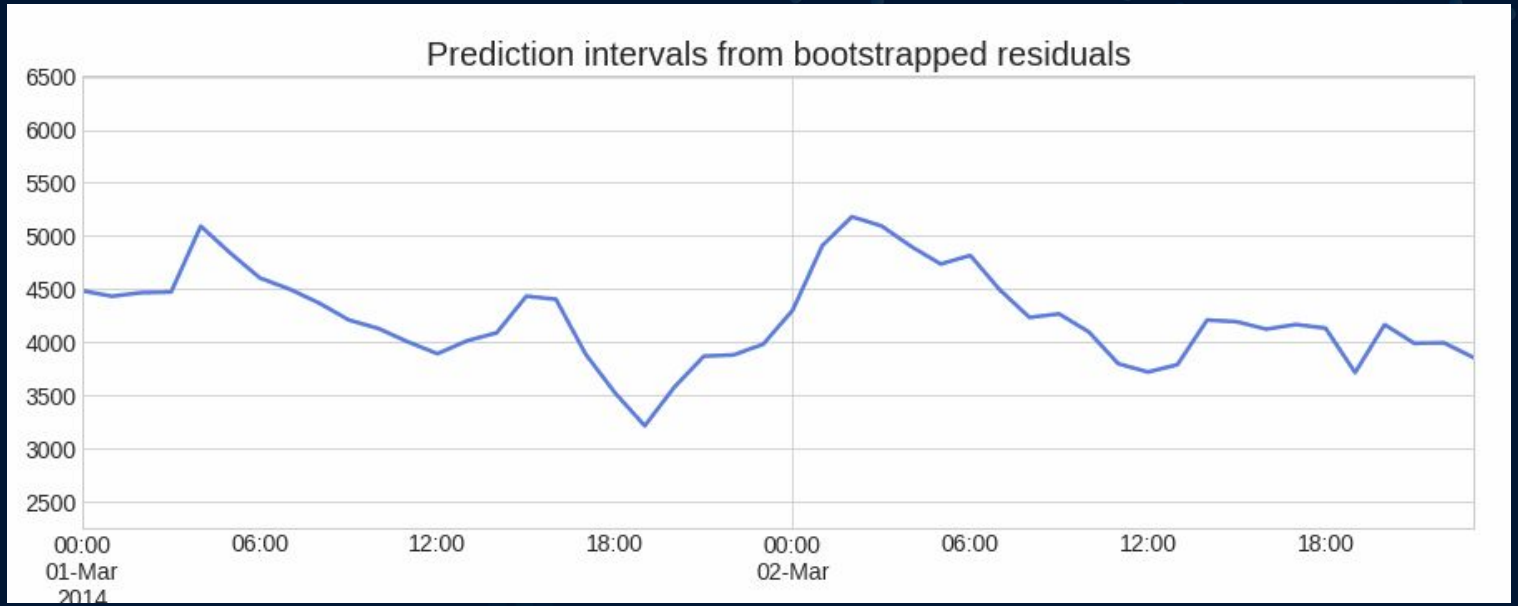
Valor predicho

Valor predicho + ϵ

Predictores

Modelo para step +1

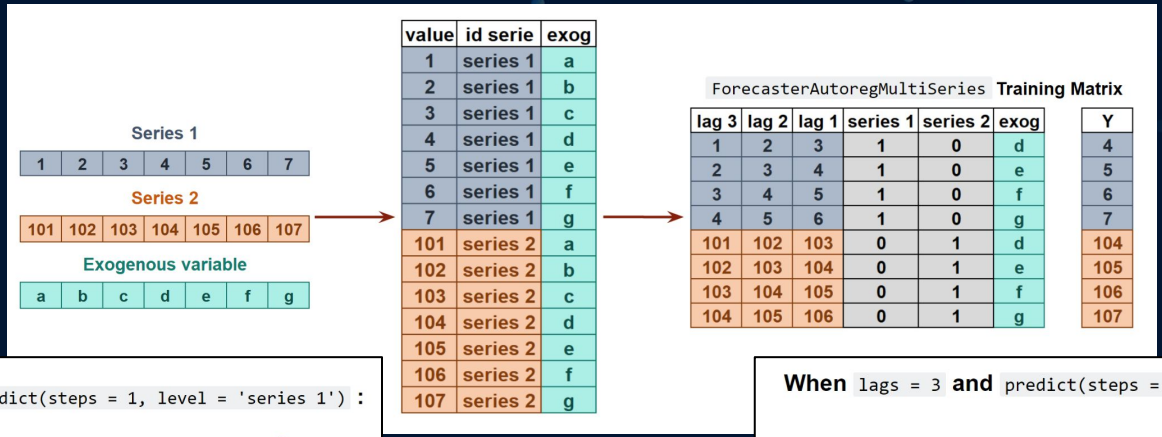
INTERVALOS DE PREDICCIÓN: BOOTSTRAPPED RESIDUALS

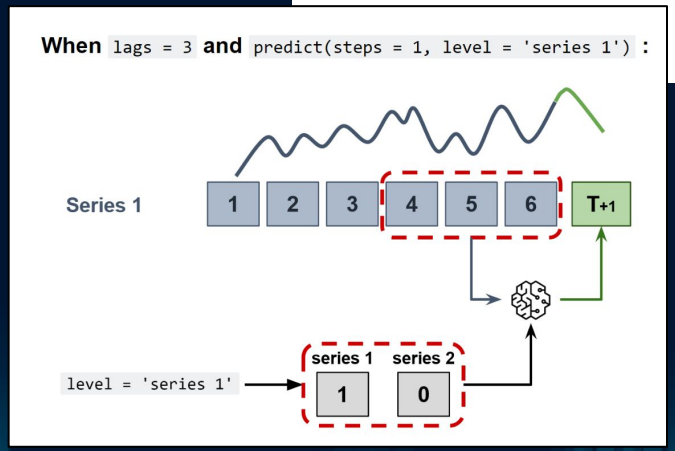


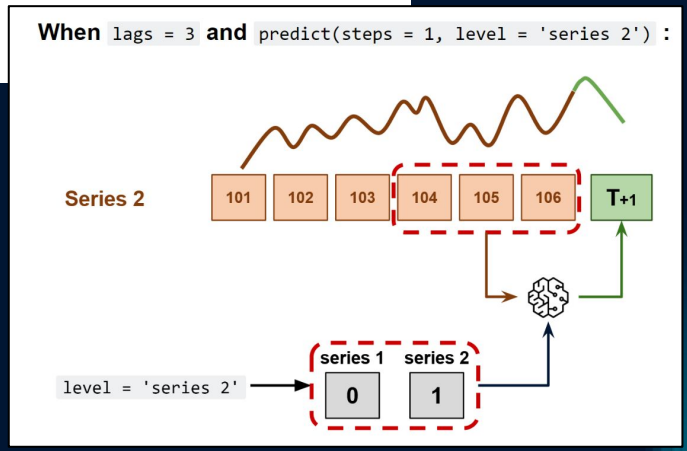
Forecaster.predict_interval()

```
# Crear y ajustar Forecaster
# ==============================================================================
forecaster = ForecasterAutoreg(
                regressor = Ridge(random_state=123),
                lags      = 12
             )

forecaster.fit(y=data_train['y'], exog=None)

# Predecir intervalo
# ==============================================================================
predicciones = forecaster.predict_interval(
                   steps    = 36,
                   interval = [10, 90] # Intervalo 80% entre percentiles 10 y 90
               )
predicciones.head(5)
```

FORECASTING MULTISERIES Y MULTIVARIATE

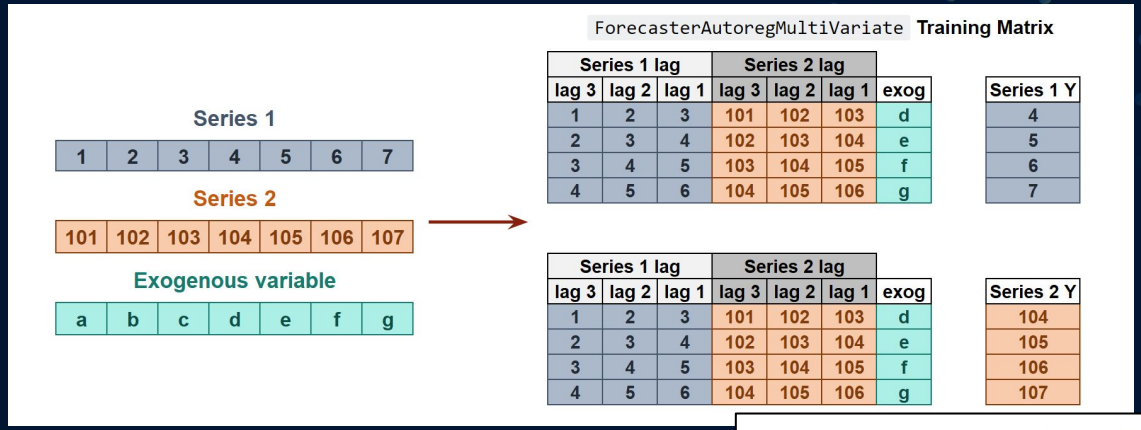


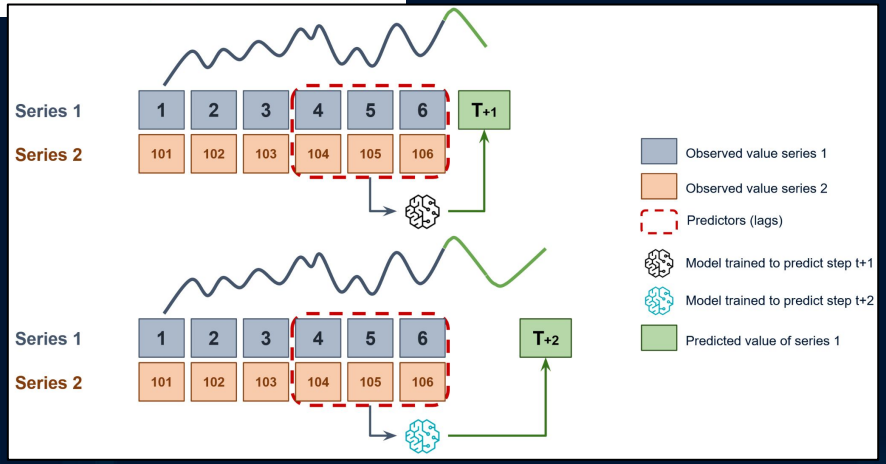




Skforecast: GitHub Skforecast: Docs www.cienciadedatos.net

FORECASTING MULTISERIES Y MULTIVARIATE





Skforecast: GitHub Skforecast: Docs www.cienciadedatos.net

ForecasterMultiSeries

```
# Crear y ajustar forecaster multi-series
# ==============================================================================
forecaster = ForecasterAutoregMultiSeries(
                regressor          = Ridge(random_state=123),
                lags               = 24,
                transformer_series = None, # posibilidad transformación individual cada serie
                transformer_exog   = None,
                weight_func        = None, # pesos observaciones series según índice
                series_weights     = None  # pesos entre series
             )

forecaster.fit(series=data, exog=None)
forecaster
```

ForecasterMultiSeries predict

```
# Predicción
# ==============================================================================
steps = 24

# Predictions for all items
predictions_items = forecaster.predict(
                        steps  = steps,
                        levels = None # levels a predecir, None==all
                    )
display(predictions_items.head(3))
print('')

# Interval predictions for item_1 and item_2
predictions_intervals = forecaster.predict_interval(steps=steps, levels=['item_1', 'item_2'])
display(predictions_intervals.head(3))
```

ForecasterMultiVariate

```
# Crear y ajustar forecaster MultiVariate
# ==============================================================================
forecaster = ForecasterAutoregMultiVariate(
                 regressor          = Ridge(random_state=123),
                 level              = 'CO', # variable objetivo del forecaster (column name)
                 lags               = 7,    # permite aplicar diferentes configuraciones de
lags para cada serie
                 steps              = 7,    # cantidad de regresores, 1 regresor por step
                 transformer_series = None,
                 transformer_exog   = None,
                 weight_func        = None
             )

forecaster.fit(series=data, exog=None)
forecaster
```

ForecasterMultiVariate predict

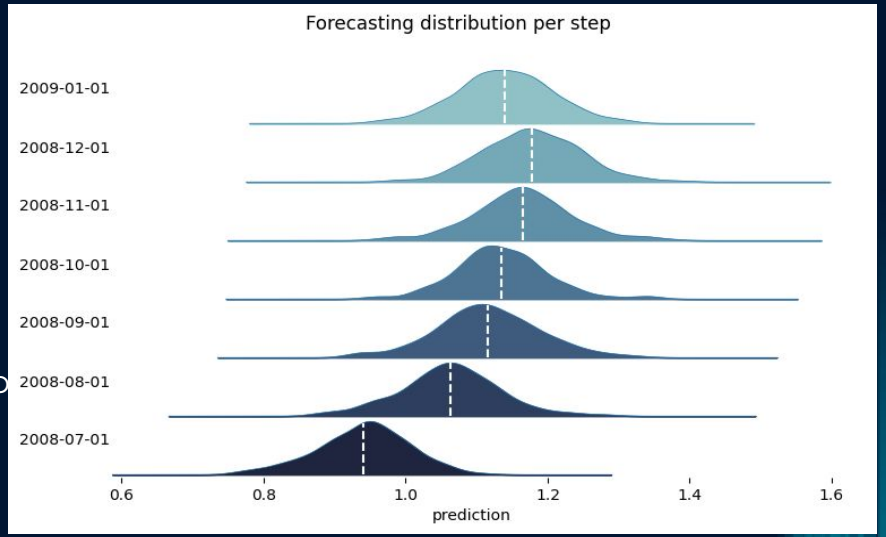
```
# Predict MultiVariate
# ==============================================================================
predicciones = forecaster.predict(steps=None) # steps a predecir
display(predicciones)
```

OTRAS FUNCIONALIDADES

- Transformación de la series temporales y variables exógenas
- Puesta en producción de modelos (last window)
- Weighted forecasting para la exclusión de partes del histórico
- Forecasting con valores nulos
- Explicabilidad de los modelos SHAP Values
- Custom metrics

Skforecast 0.7.0 new release?

- Predicción de distribuciones
- SARIMAX wrapper
- Predicción de intervalo en la estrategia direct



Skforecast: GitHub Skforecast: Docs www.cienciadedatos.net

MATERIAL ADICIONAL

- **Casos prácticos (link)**
  - Skforecast: forecasting series temporales con Python y Scikit-learn
  - Forecasting de la demanda eléctrica
  - Forecasting de las visitas a una página web
  - Forecasting series temporales con gradient boosting: Skforecast, XGBoost, LightGBM y CatBoost
  - Predicción del precio de Bitcoin con Python
  - Intervalos de predicción en modelos de forecasting
  - Multi-series forecasting

- **Documentación**
  - Github: Skforecast
  - Documentación y guías de usuario Skforecast

OTRAS LIBRERÍAS DE FORECASTING EN PYTHON

- **sktime**
- **StatsForecast (Nixtla)**
- **Prophet**
- **NeuralProphet**

NECESITAMOS VUESTRA AYUDA 😊

- Feedback a través de Github issues
- **Estrellita en GitHub ⭐**
- Sugerir mejoras
- Recomendarla a tus amigos data scientists

¿PREGUNTAS?

Joaquín Amat Rodrigo
Javier Escobar Ortiz