A Crash course in Time Series Forecasting from Naive to Foundational

Pietro Peterlongo, Data Scientist @ AgileLab

Agenda

1. why forecasting (and nixtla)
2. minimal example (and statsforecast)
3. more (M5, ml, hierarchical, neural, foundational)

👋 Hi, I am Pietro 👨‍👩‍👧

- Data Scientist @ agilelab.it ⚪🔵
  - 📘handbook, 🤙holacracy (self-management)
- (previously) 8+ years at Software Vendor in Supply Chain
- 🐍🇮🇹 Python/PyData Milano milano.python.it
- 🐙 github.com/pietroppeter (👈 more socials 🔗)

🎪 Come to PyCon Italy (Bologna, May 29-31)! 🍝

why forecasting?

domains

- 📈 sales/demand
- 🔋 energy consumption
- 💹 financial assets
- 🌤️ weather
- ...

where is the (business) value? 💰

take better decisions! 💡

🏹 taking time seriously

time is the most important dimension

- time frequency (months, weeks, days, hours, …)
- time horizon (how many weeks in the future)?
- lag n forecast: the forecast for the n-th time bucket in the future

other dimensions (e.g. product, market, …) usually lead to forecasting *multiple* time series (or *multivariate* ones)

nixtla

StatsForecast

Lightning fast forecasting with statistical and econometric models.

MLForecast

Scalable machine learning for time series forecasting.

NeuralForecast

Scalable and user friendly neural forecasting algorithms for time series data.

Hierarchical Forecast

Probabilistic Hierarchical forecasting with statistical and econometric methods.

TS features

Calculates various features from time series data. Python implementation of the R package ts features.

Nixtla/Nixtla

Offers a collection of classes and methods to interact with the API of TimeGPT.

methodology

1. think about your why
2. gather data (process, explore)
3. baseline
4. measure
5. improve
6. restart from step 4 or less

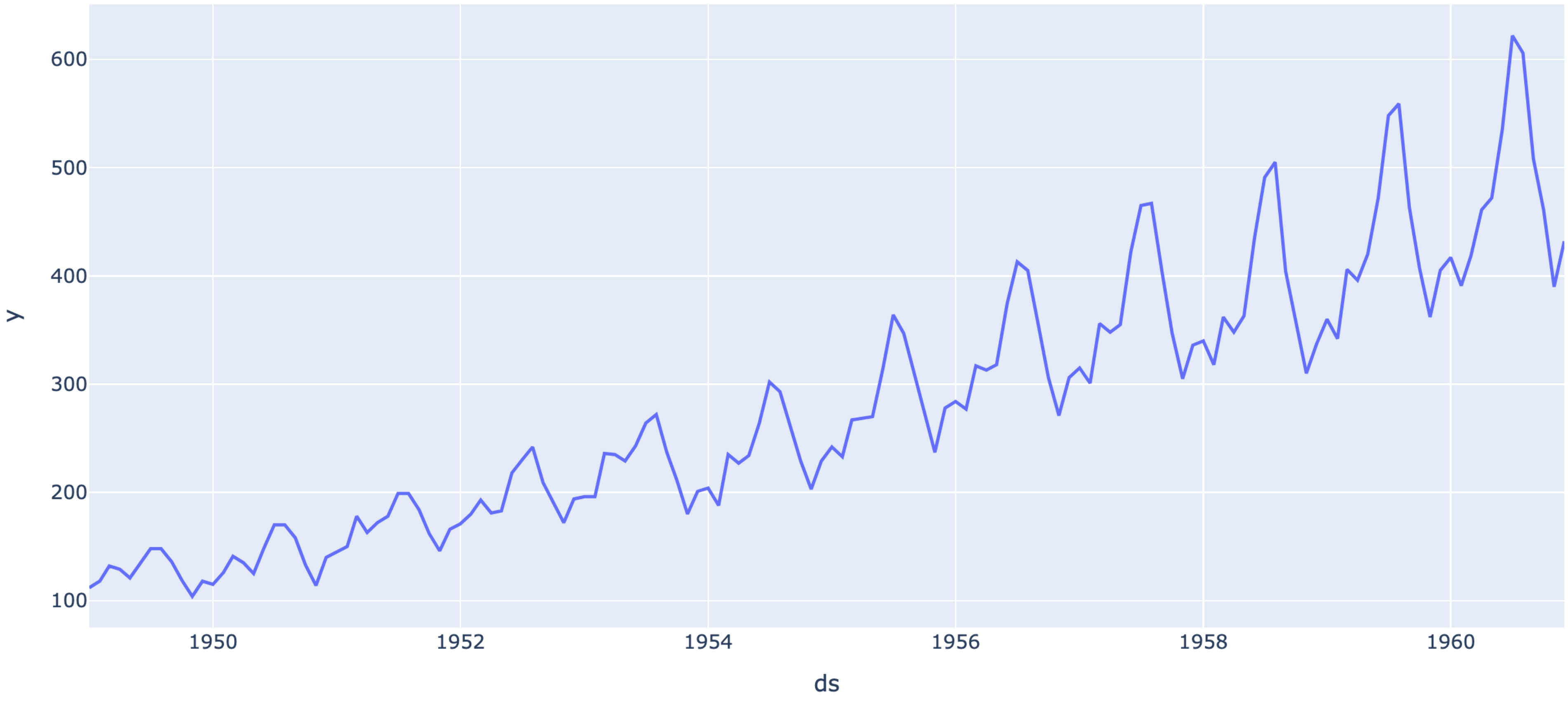
AirPassengers - data

```python
import polars as pl

url = "https://datasets-nixtla.s3.amazonaws.com/air-passengers.csv"
air = pl.read_csv(url).with_columns(pl.col("ds").str.to_date())
air.head(3)
```

unique_id	ds	y
str	date	i64
"AirPassengers"	1949-01-01	112
"AirPassengers"	1949-02-01	118
"AirPassengers"	1949-03-01	132

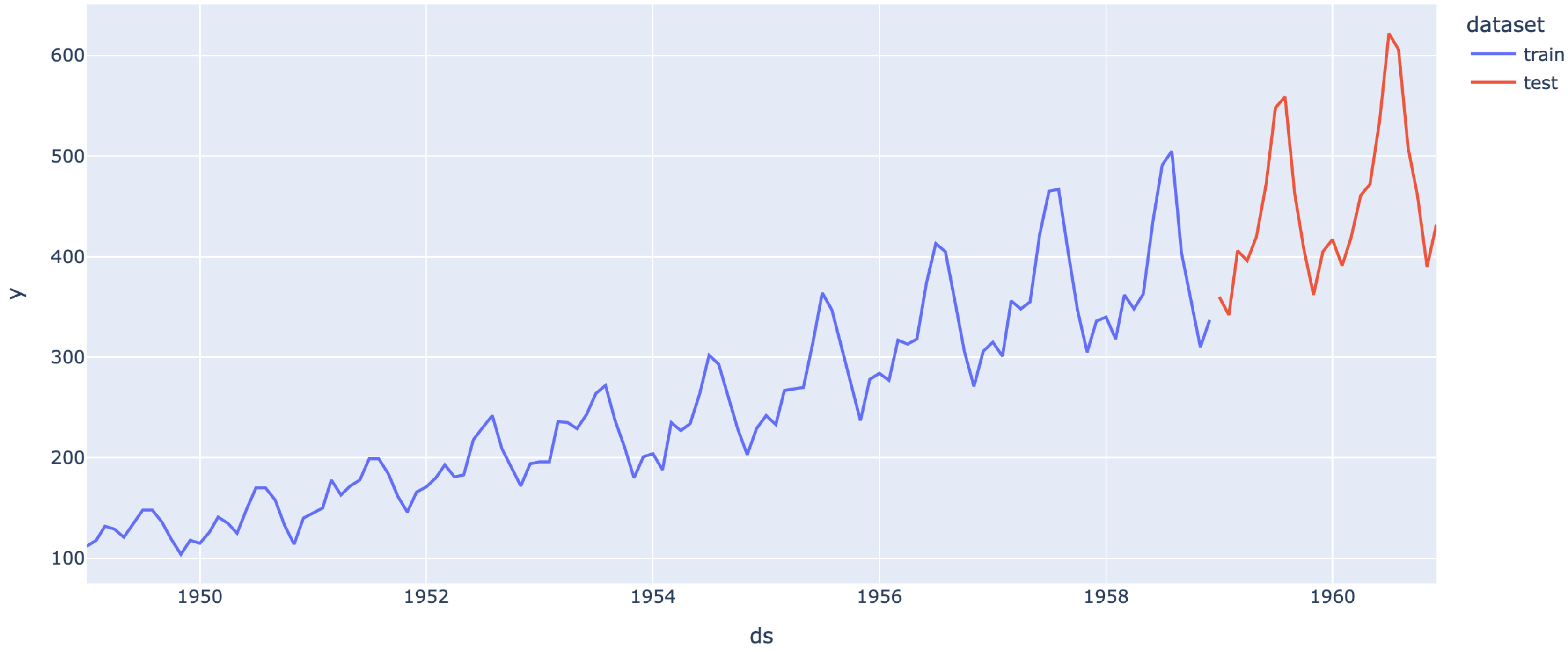
AirPassengers - plot

```
import plotly.express as px

px.line(air, x="ds", y="y")
```



split train/test

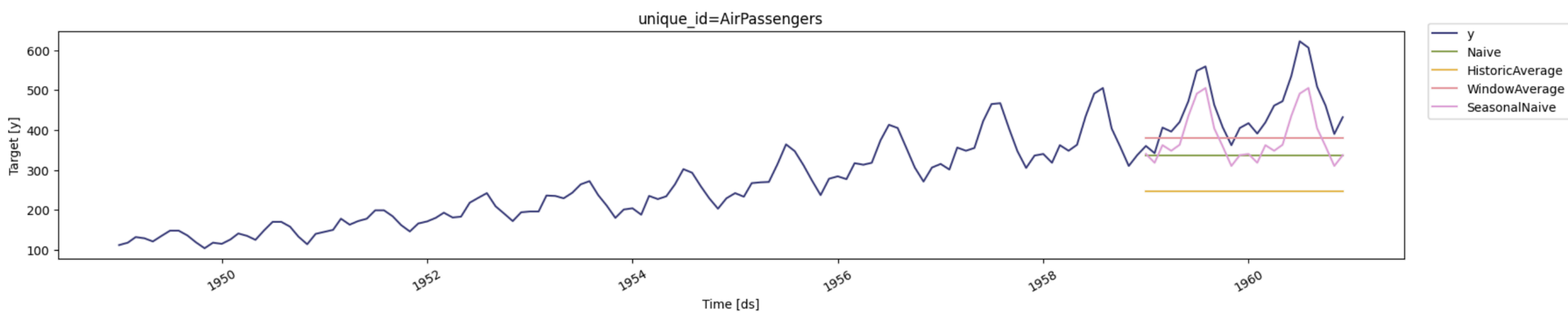
```python
from datetime import date

cutoff = date(1959, 1, 1)
df = air.with_columns(pl.when(pl.col("ds") < cutoff).then(pl.lit("train"))
  .otherwise(pl.lit("test")).alias("dataset"))
px.line(df, x="ds", y="y", color="dataset")
```



agilelab

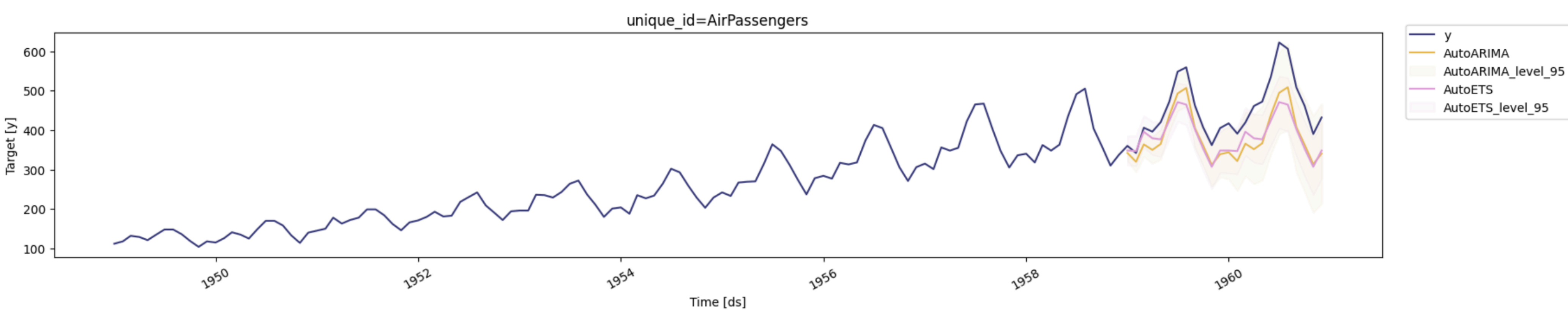
baseline

```python
from statsforecast import StatsForecast
from statsforecast.models import Naive, HistoricAverage, WindowAverage, Sea

models=[Naive(),HistoricAverage(), WindowAverage(window_size=12),
        SeasonalNaive(season_length=12)]
sf = StatsForecast(models=models, freq="MS")
sf.fit(train_df)
predict_df = sf.predict(h=24)
sf.plot(df, predict_df)
```



statsforecast

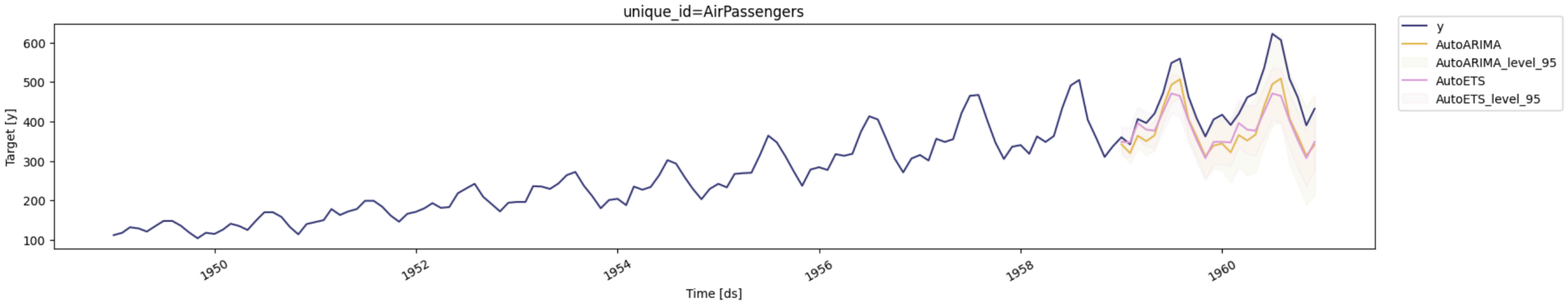
```python
from statsforecast.models import AutoARIMA, AutoETS

sf = StatsForecast(
    models=[
        AutoARIMA(season_length=12),
        AutoETS(season_length=12),
    ], freq="MS")
sf.fit(train_df)
predict_df = sf.predict(h=24, level=[95])
sf.plot(df, predict_df, level=[95])
```



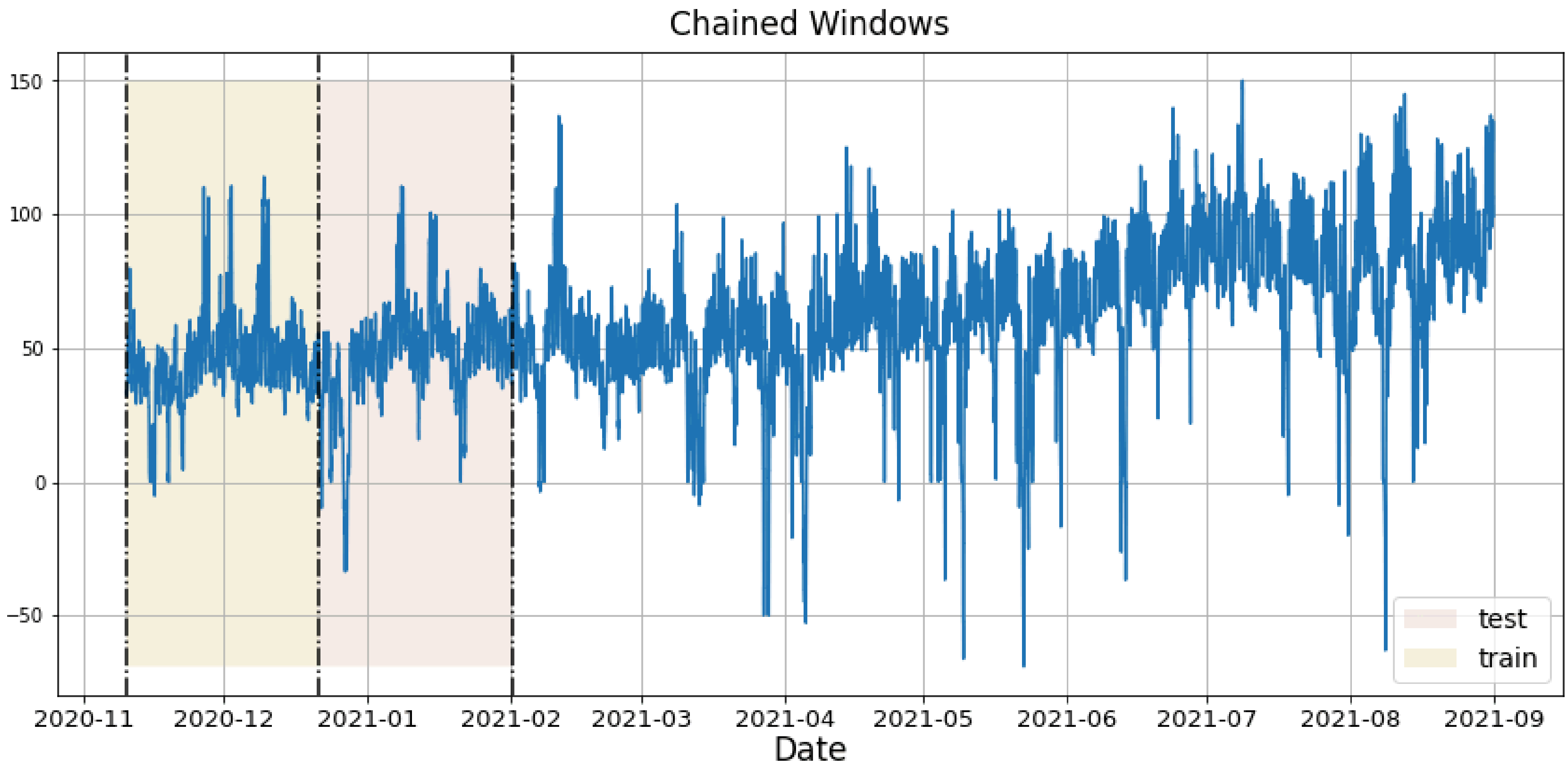
probabilistic forecast

Note that Nixtla provides for (almost) all its models a probablistic forecast (using `levels` keyword argument), either through model specific estimates or with **conformal prediction** (model agnostic)

```
predict_df = sf.predict(h=24, level=[95])
sf.plot(df, predict_df, level=[95])
```



cross validation



measure

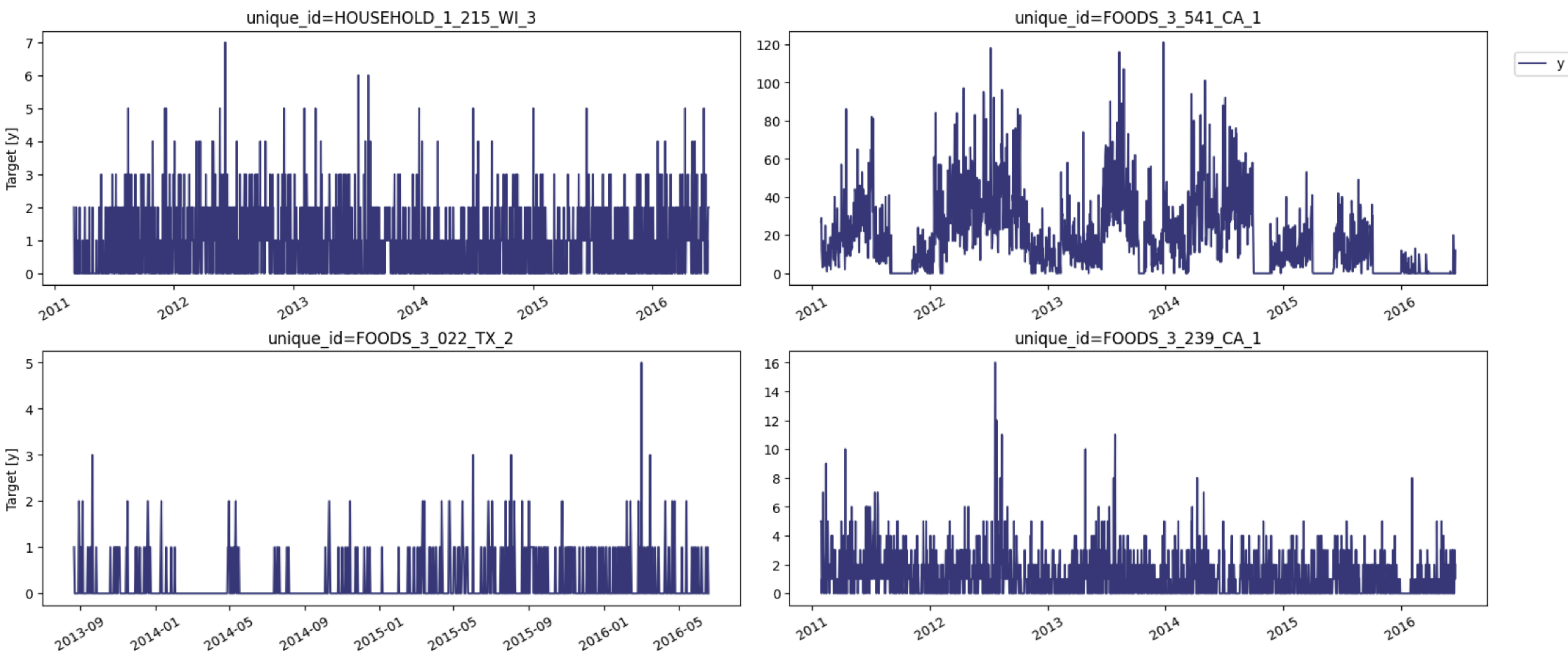
```python
from utilsforecast.losses import rmse

cv_df = sf.cross_validation(df = df, h = 24, step_size = 24, n_windows = 3)
rmse(cv_df, models=["AutoARIMA", "AutoETS"])
```

	unique_id	AutoARIMA	AutoETS
0	AirPassengers	55.872734	56.214554

agilelab

M5 Forecasting competition

```python
from datasetsforecast.m5 import M5
Y_df, X_df, S_df = M5.load("data")
sf.plot(Y_df)
```



mlforecast

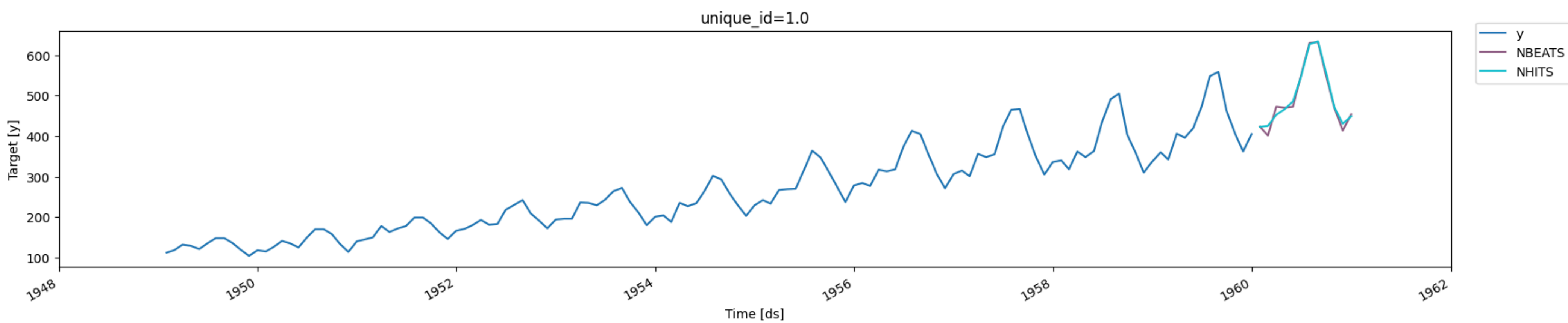
```python
import lightgbm as lgbm
from mlforecast import MLForecast
from mlforecast.lag_transforms import ExpandingMean, RollingMean
from mlforecast.target_transforms import Differences

fcst = MLForecast(
    models=[lgbm.LGBMRegressor()],
    freq='D',
    lags=[7, 14],
    lag_transforms={
        1: [ExpandingMean()],
        7: [RollingMean(window_size=28)]
    },
    date_features=['dayofweek'],
    target_transforms=[Differences([1])],
)
```

hierarchical forecast

```python
from datasetsforecast.hierarchical import HierarchicalData
from hierarchicalforecast.core import HierarchicalReconciliation
from hierarchicalforecast.methods import BottomUp, TopDown, MiddleOut

# Create timeseries for all levels of the hierarchy
Y_df, S, tags = HierarchicalData.load('./data', 'TourismSmall')
# ...
Y_train_df, Y_test_df = ...

# Compute base predictions
fcst = StatsForecast(models=[AutoARIMA(season_length=4), freq='QE')
Y_hat_df = fcst.forecast(df=Y_train_df, h=4)

# Reconcile the base predictions
reconcilers = [
    BottomUp(),
    TopDown(method='forecast_proportions'),
    MiddleOut(middle_level='Country/Purpose/State',
```

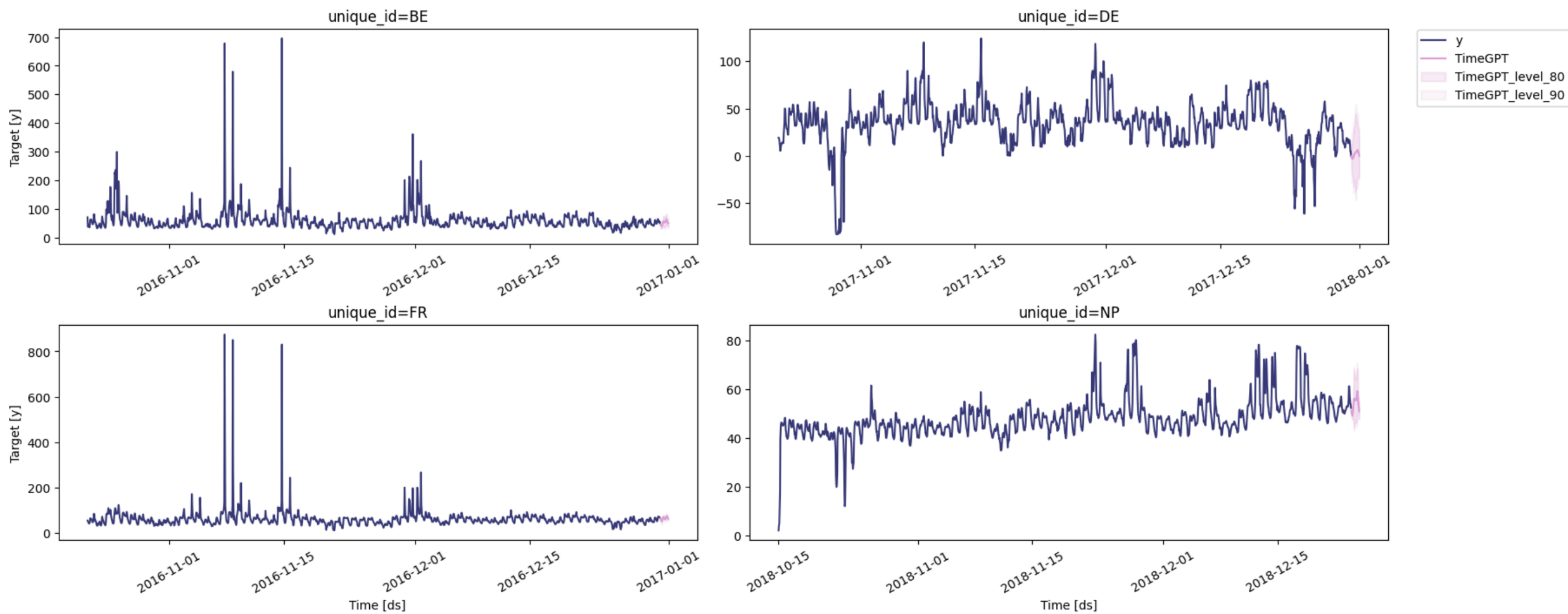
neural forecast

```python
from neuralforecast import NeuralForecast
from neuralforecast.auto import AutoNHITS, AutoLSTM
# ...
horizon = len(Y_test_df)
models = [NBEATS(input_size=2 * horizon, h=horizon, max_steps=100),
          NHITS(input_size=2 * horizon, h=horizon, max_steps=100)]
nf = NeuralForecast(models=models, freq='ME')
nf.fit(df=Y_train_df)
Y_hat_df = nf.predict()

plot_series(Y_df, Y_hat_df)
```



foundational models

```python
from nixtla import NixtlaClient

nixtla_client = NixtlaClient(api_key = nixtla_api_key)
df = pd.read_csv('https://raw.githubusercontent.com/Nixtla/transfer-learnin
fcst_df = nixtla_client.forecast(df, h=24, level=[80, 90])
nixtla_client.plot(df, fcst_df, level=[80, 90])
```



🙏 Thank you for listening!