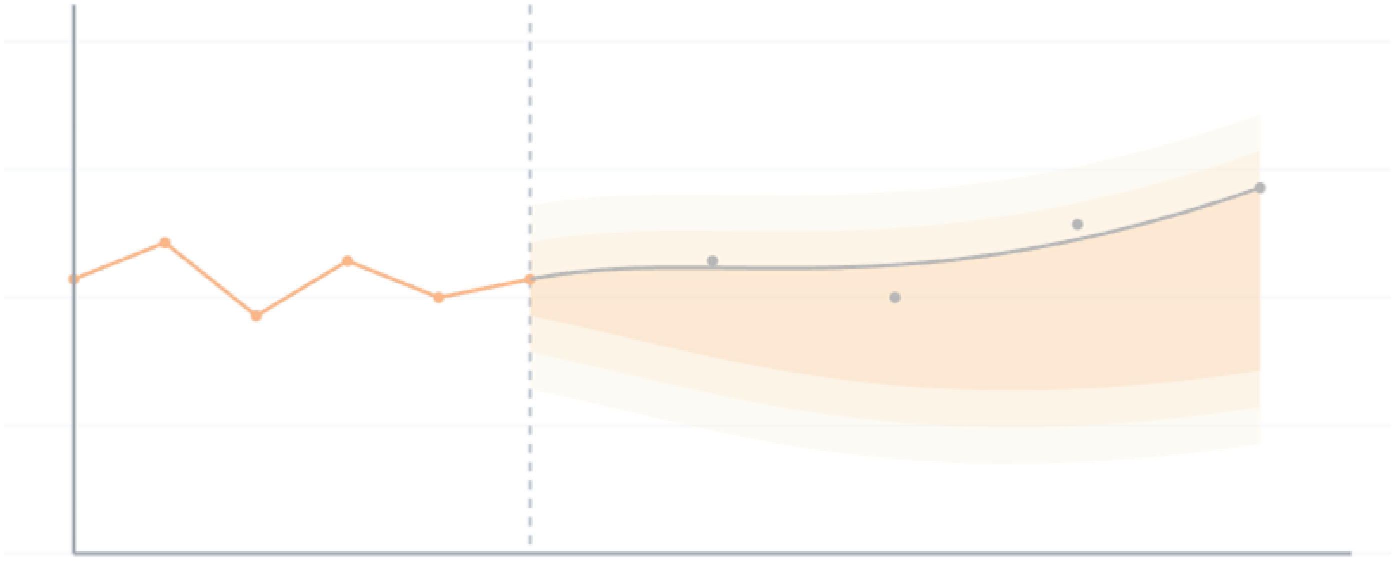
PyData NYC 2024

Udisha Dutta Chowdhury

**Joint work with Abhishek Murthy,
Schneider Electric**



Adopting Open-Source Tools for Time Series Forecasting: Opportunities and Pitfalls

About Me

Udisha Dutta Chowdhury

- Master’s Student in **Computer System’s Engineering** at **Northeastern University**, Boston, MA.
- **Teaching Assistant** for Machine Learning for IoT Systems Course at **Northeastern University.**
- **Data Science Intern** at **Schneider Electric**, Andover, MA.
- Cyber Security Intern and Solution Delivery Analyst at **Deloitte USI**, Bangalore, India.
- Undergrad in **Electronics and Communication Engineering, PES University,** Bangalore, India.

Outline of the Talk

Introduction to Time Series Forecasting

Open-Source Tools

3 critical aspects of forecasting libraries

Data Understanding

Data Preparation

Backtesting

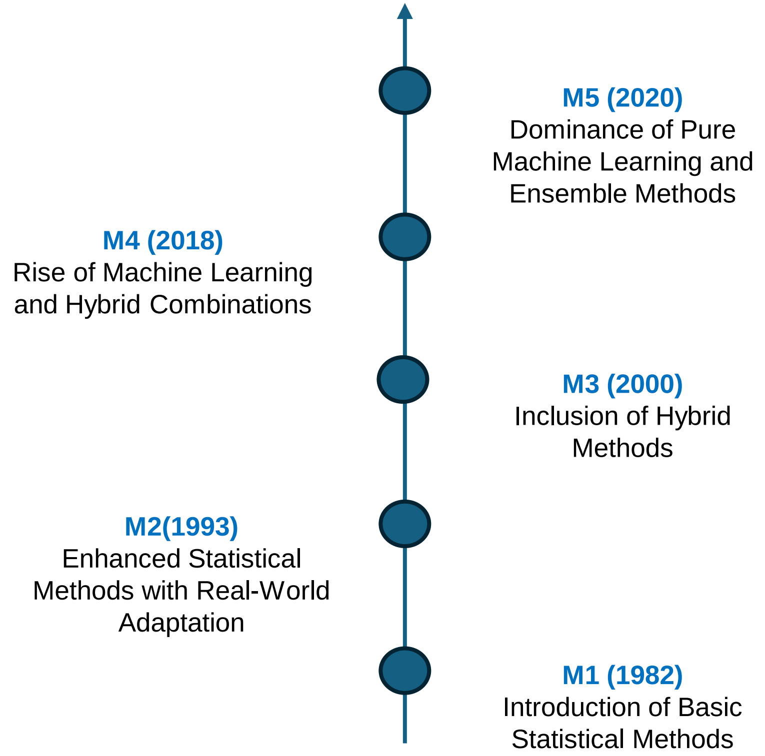
Conclusion and Key Takeaways

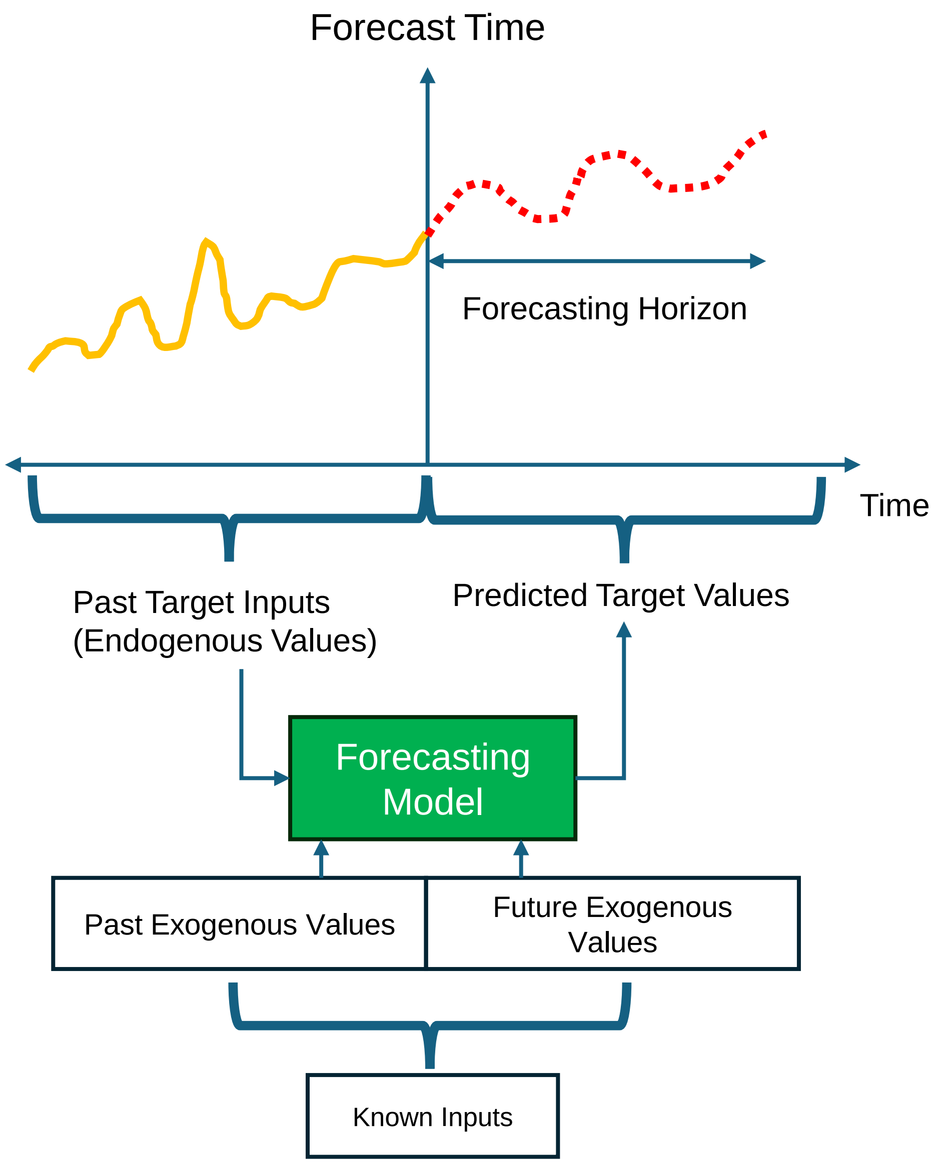
Time Series Forecasting

Evolution of Forecasting Models

M Competitions benchmark forecasting models, showcasing advancements from traditional statistical methods to modern machine learning approaches

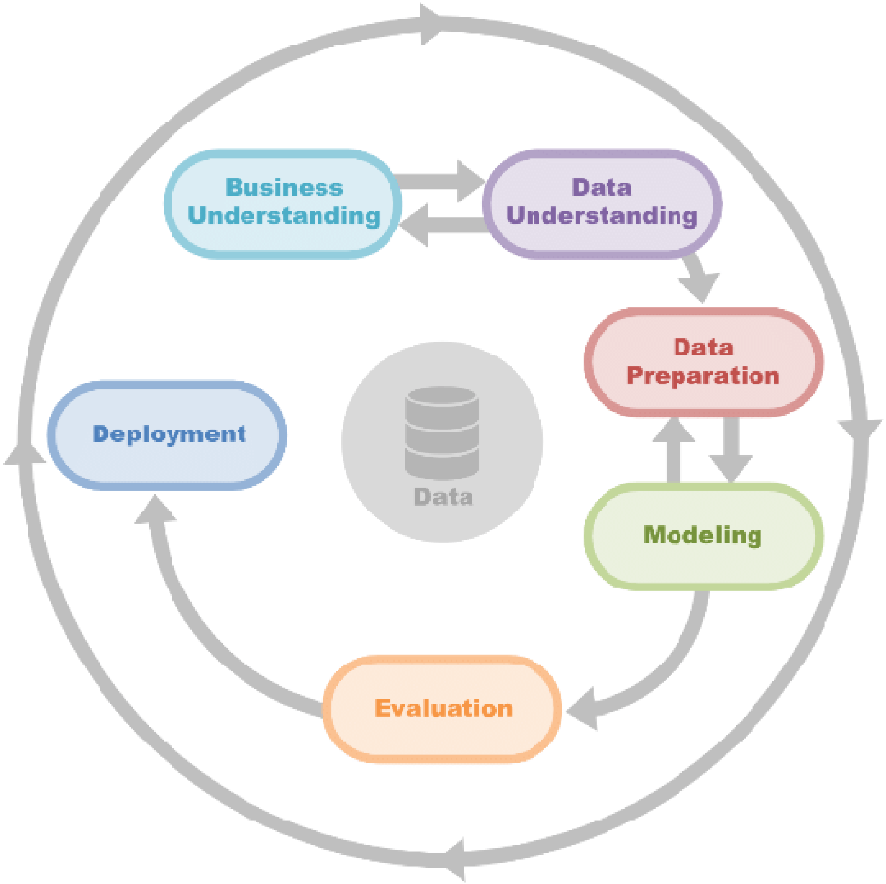
Milestones in the M Competitions





Common Stages in a Forecasting Project

- **Business Understanding:**
  - Define the forecasting objective: What do we need to predict, and why is it important?
- **Data Understanding:**
  - Explore time-series data: Identify trends, seasonality, and patterns.
- **Data Preparation:**
  - Preprocess data for forecasting: Handle missing values, NaNs, duplicate values.
- **Modeling:**
  - Apply forecasting methods: Choose models (e.g., ARIMA, neural networks) based on data characteristics.
- **Evaluation (Backtesting):**
  - Test model accuracy under realistic settings.
- **Deployment:**
  - Rollout the model in production.

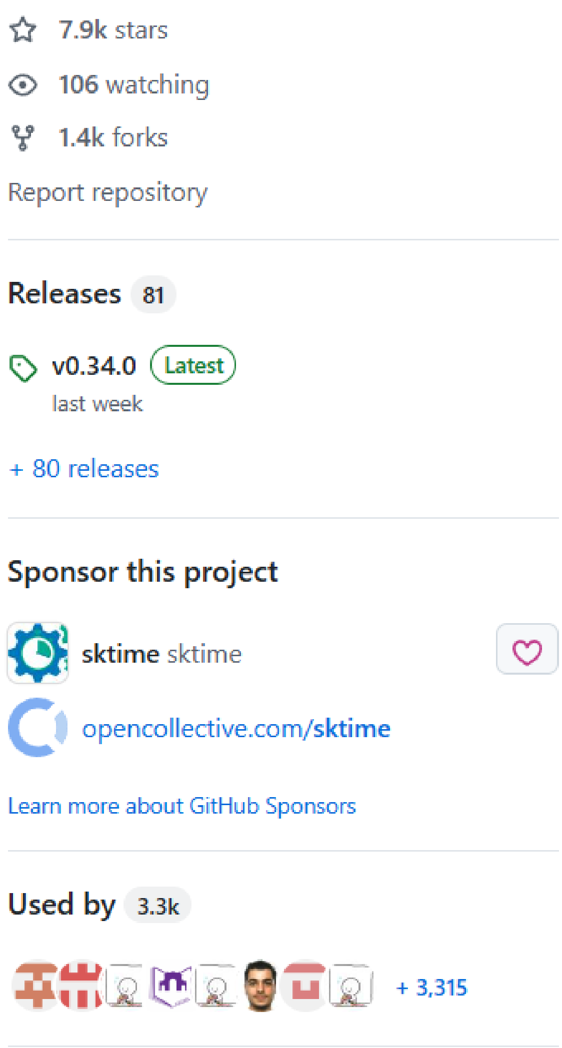


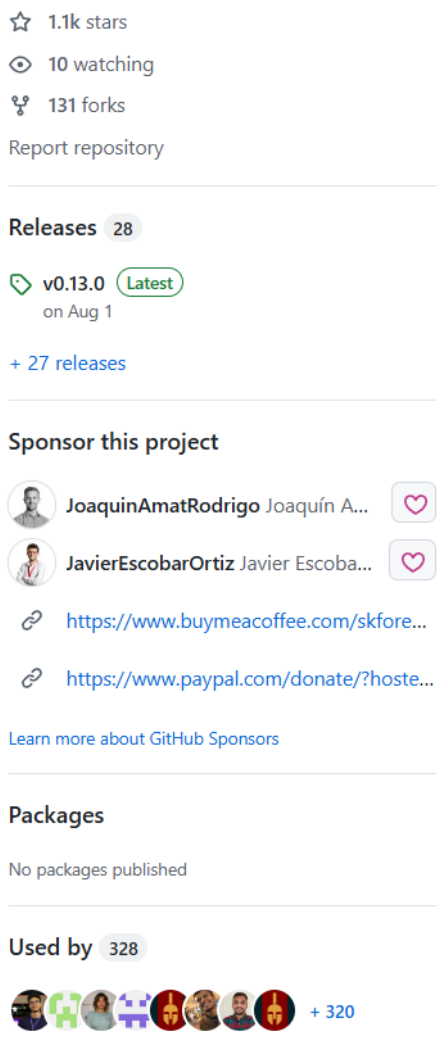
Open-Source Tools for Time Series Forecasting

Advantages of Open-Source Solutions

- Accessibility and Cost-Effectiveness
- Flexibility and Customizability
- Community-Driven Innovation
- Integration with Existing Ecosystems

Growing Popularity of SKtime and Skforecast





- GitHub Stars and Repository Activity
- Community Engagement and Usage Trends

3- Critical Aspects for Forecasting Libraries

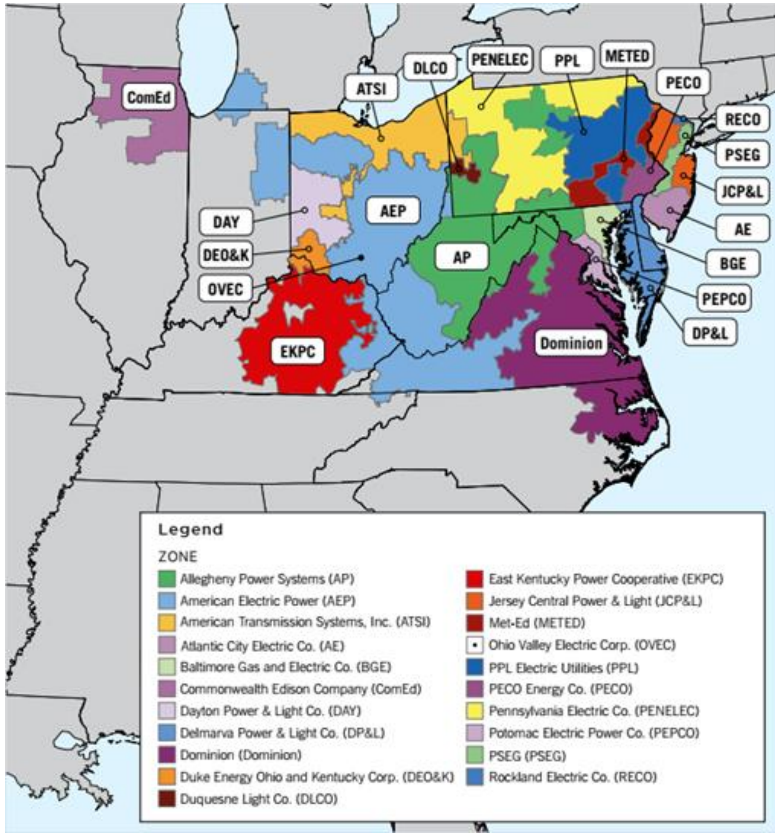
Data Understanding: How well does the library support Exploratory Data Analysis (EDA)?

Data Preparation: How robust and intuitive are the tool's preprocessing capabilities for handling quality issues, like missing values, NaNs, duplicate data, and exogenous variables?

Backtesting: How effective and scalable are the library’s modeling and evaluation capabilities for forecasting algorithms?

Dataset used for the experiments

PJM Hourly Energy Consumption Data



- Independent nonprofit organization that manages the electric transmission system for a large region
- Hourly power consumption data comes from PJM's website and are in megawatts (MW).
- Data used : 125 days of hourly power consumption of Duquesne Light Company (DUQ)

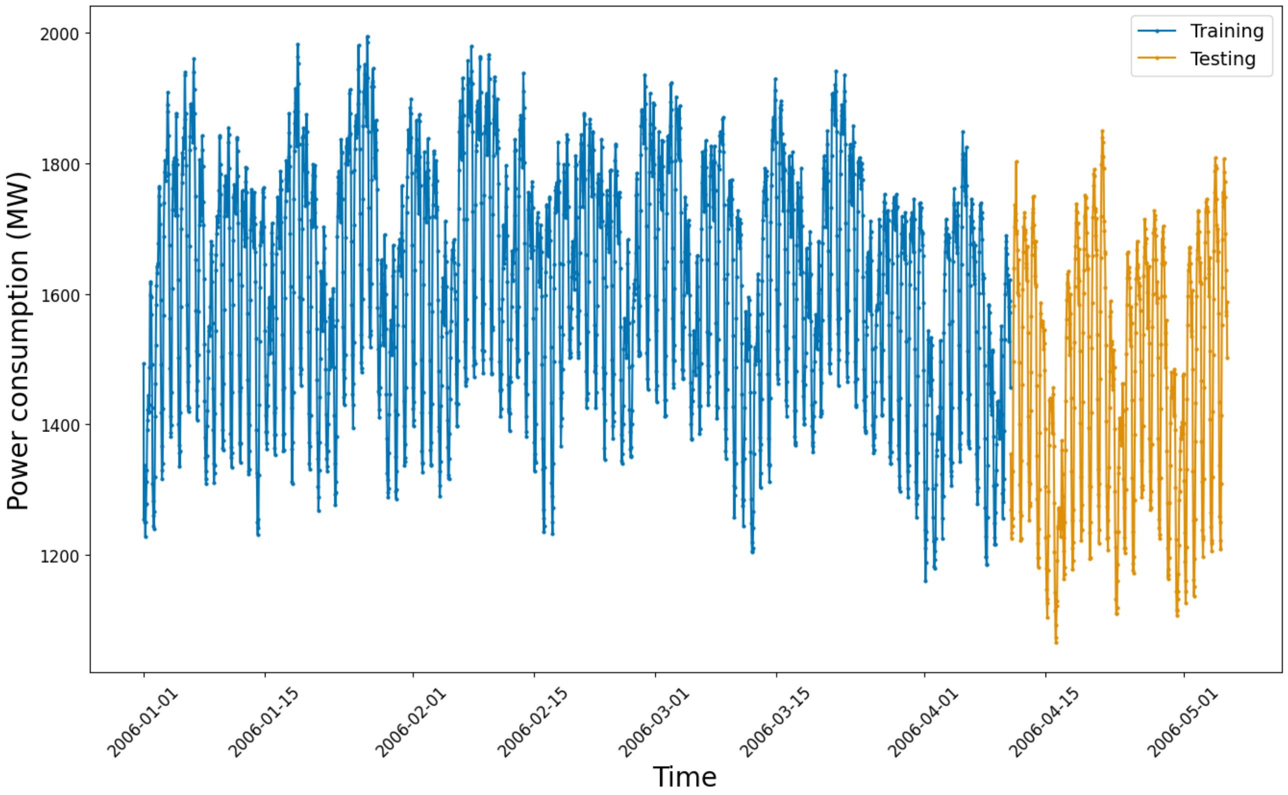
Data Understanding : Forecasting Libraries accelerate EDA

SKTIME

Viz. trends with train-test split

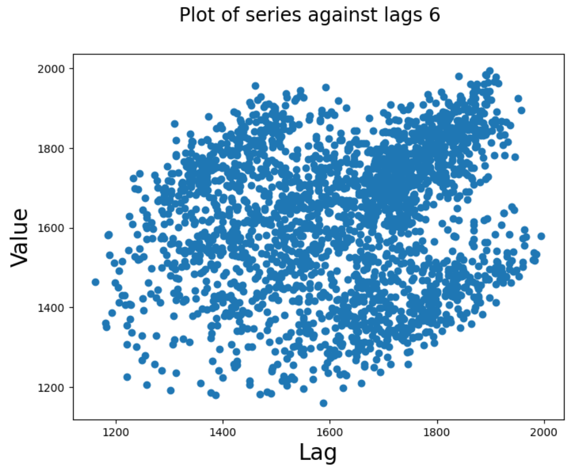
`plot_series()`

```
plot_series(y_train, y_test, labels=["Training", "Testing"], markers=['.', '.'], ax=ax)
```

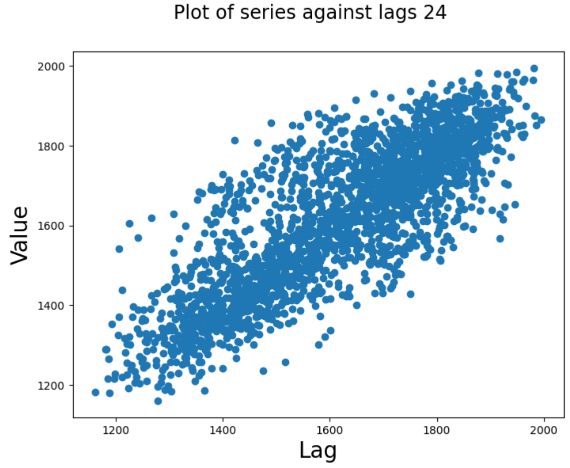


Plot one or more lagged versions of the time-series data

`plot_lags()`




```
plot_lags(y_train, lags=6)
```



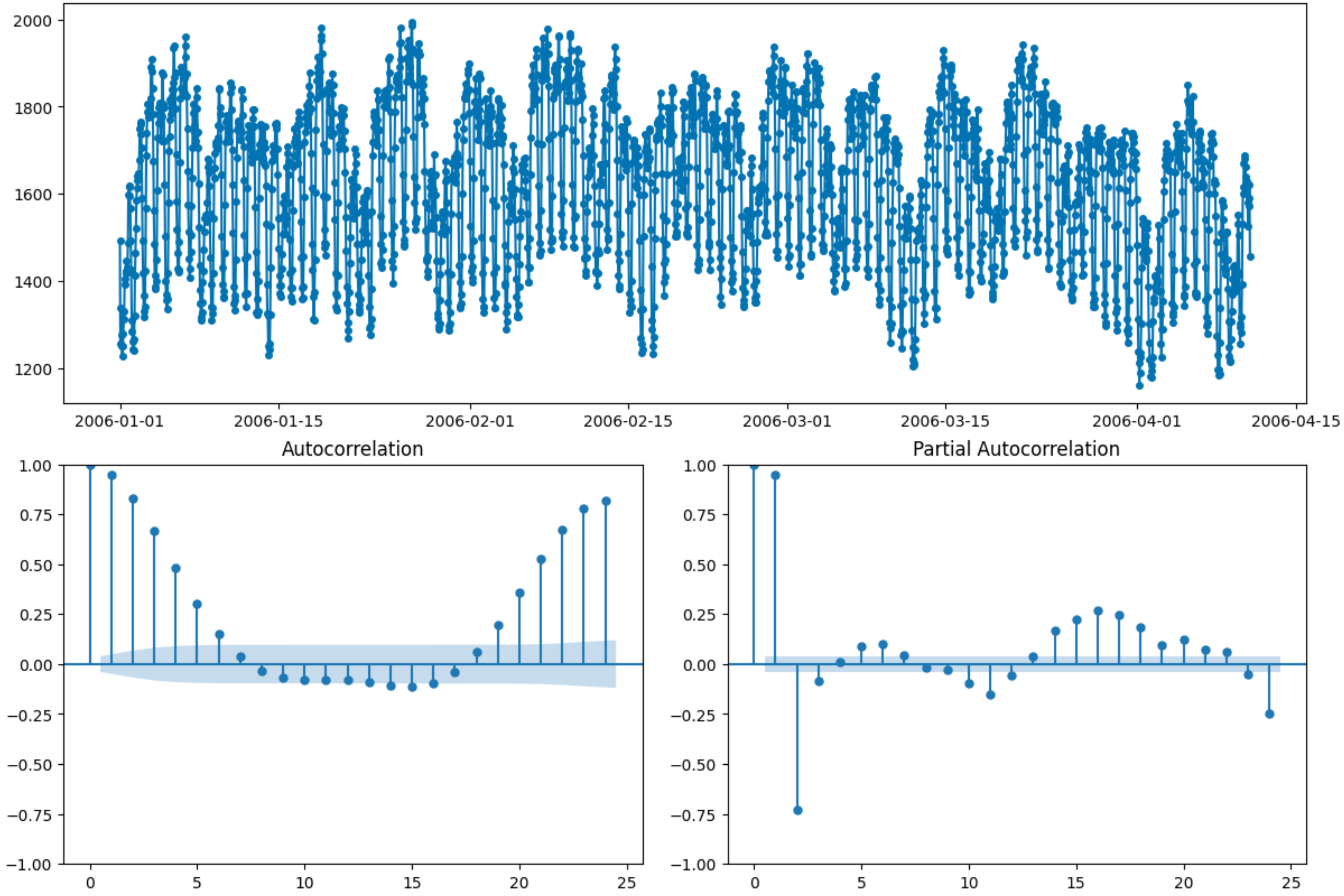

```
plot_lags(y_train, lags=24)
```

Data Understanding

Correlation (auto & partial)

`plot_correlations()`

`plot_correlations(y_train)`



Data Preparation: Forecasting Libraries offer pre-processing features

time	y
t_0	y_0
$t_0 + \delta$	y_1
$t_0 + 2\delta$	y_2
$t_0 + 3\delta$	y_3
$\vdots$	

Nominal

time	y
t_0	y_0
$t_0 + \delta$	y_1
$t_0 + 2\delta$	NaN
$t_0 + 3\delta$	y_3
$\vdots$	

Missing data

time	y
t_0	y_0
$t_0 + \delta$	y_1
$t_0 + 3\delta$	y_3
$t_0 + 4\delta$	y_4
$\vdots$	

Irregular data

time	y
t_0	y_0
$t_0 + \delta$	y_1
$t_0 + \delta$	y_1
$t_0 + 2\delta$	y_2
$\vdots$	

Duplicate data

Data quality issues are expected in the training and testing datasets.

The forecasting library must help the data scientist analyze and mitigate the issues through:

- Missing value imputation
- Outlier detection and removal
- Detrending and seasonality adjustment
- Resampling irregular timestamps to fixed frequency
- Time-based train-test splitting utilities

Data Preparation

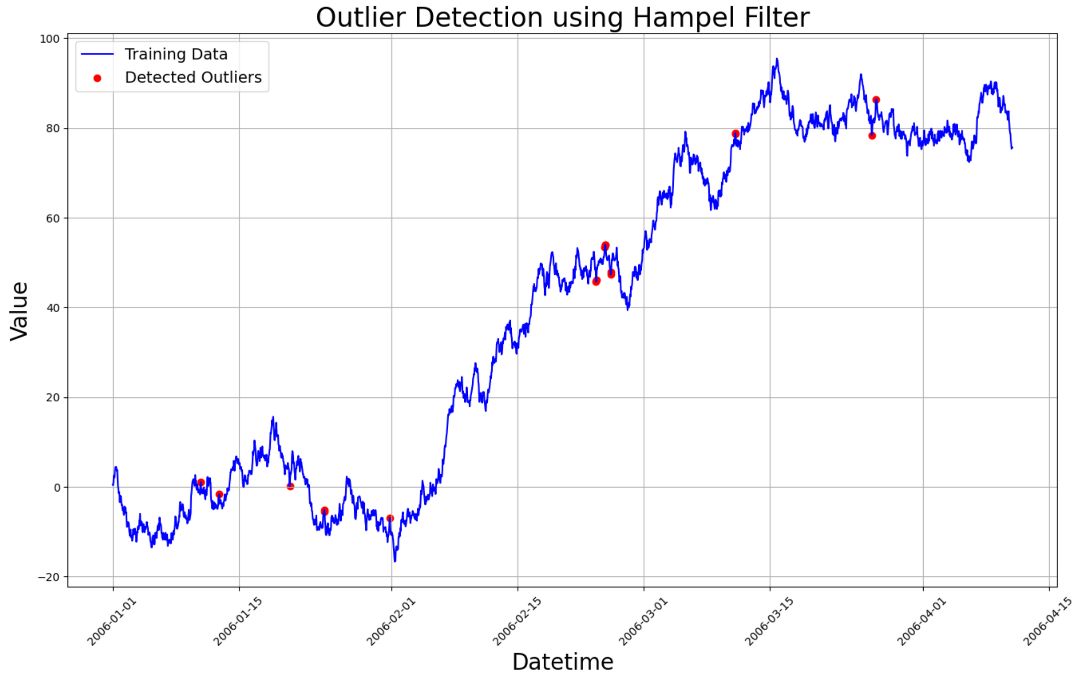
Missing Value Imputation

```
from sktime.transformations.series.impute import Imputer

imputer = Imputer(method="mean")
data_imputed = imputer.fit_transform(y_train)
```

```
Original Data with Missing Values:
2006-01-01 00:00:00    0.496714
2006-01-01 01:00:00         NaN
2006-01-01 02:00:00    1.006138
2006-01-01 03:00:00         NaN
2006-01-01 04:00:00    2.295015
Freq: H, Name: Value, dtype: float64

Imputed Data:
2006-01-01 00:00:00     0.496714
2006-01-01 01:00:00    50.037261
2006-01-01 02:00:00     1.006138
2006-01-01 03:00:00    50.037261
2006-01-01 04:00:00     2.295015
Freq: H, Name: Value, dtype: float64
```

Outlier Detection and Removal

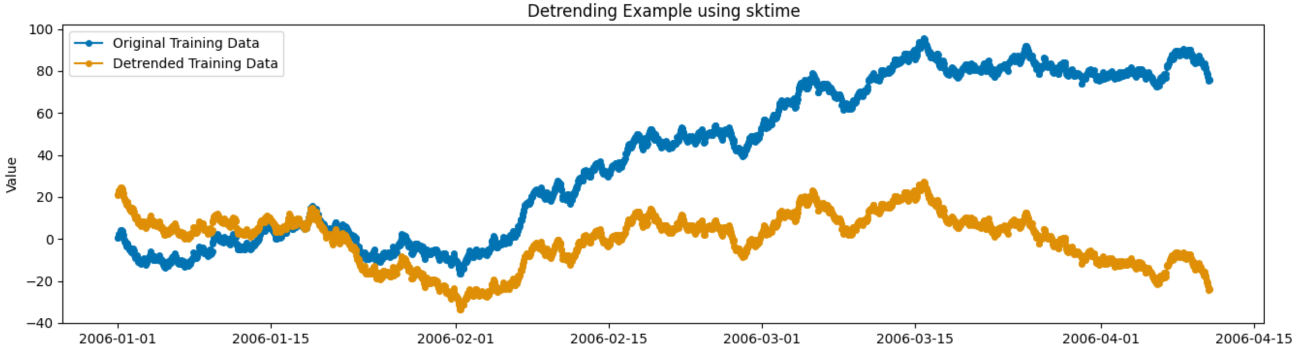
```
from sktime.transformations.series.outlier_detection
import HampelFilter

hampel_filter = HampelFilter(window_length=24,
n_sigma=3)
outliers = hampel_filter.fit_transform(y_train)
```



Data Preparation

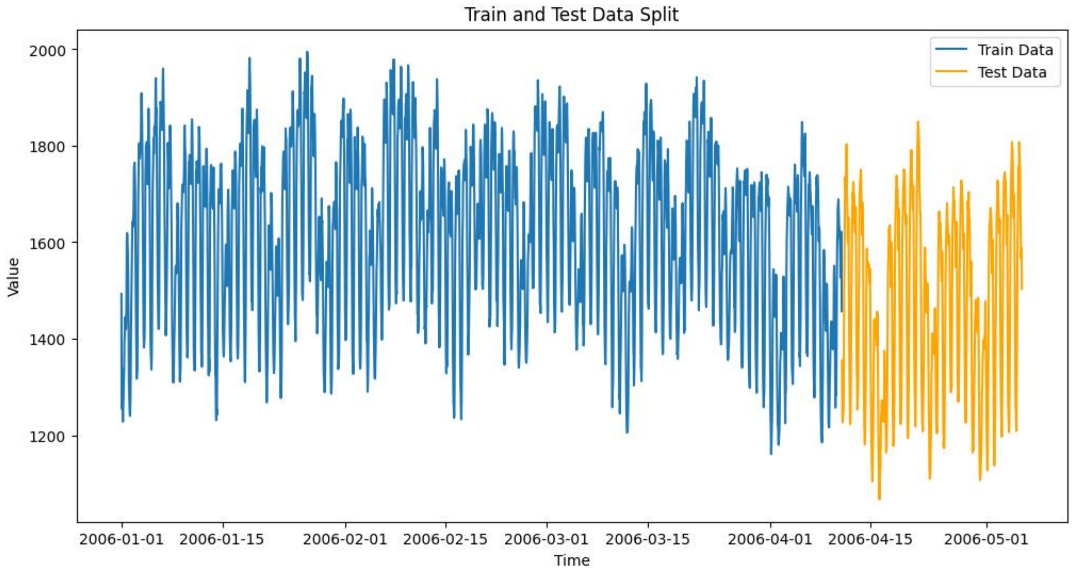
Detrending and Seasonality Adjustment

```
from sktime.transformations.series.detrend import
Detrender

detrender = Detrender()
y_train_detrended = detrender.fit_transform(y_train)
```



Time-based train-test split

```
from sktime.split import temporal_train_test_split

data = df['2006-01-01':'2006-12-31'][:(24*125)]
y_train, y_test =
         temporal_train_test_split(data, test_size=0.2)
```



Data Preparation

Skforecast works with pandas for data preprocessing

Handle Outliers using Rolling Statistics

```
roll_mean = data_clean.rolling(window=5, center=True).mean()
roll_std = data_clean.rolling(window=5, center=True).std()
```

Resampling Irregular Timestamps

```
data_resampled = data.resample('H').mean()
```

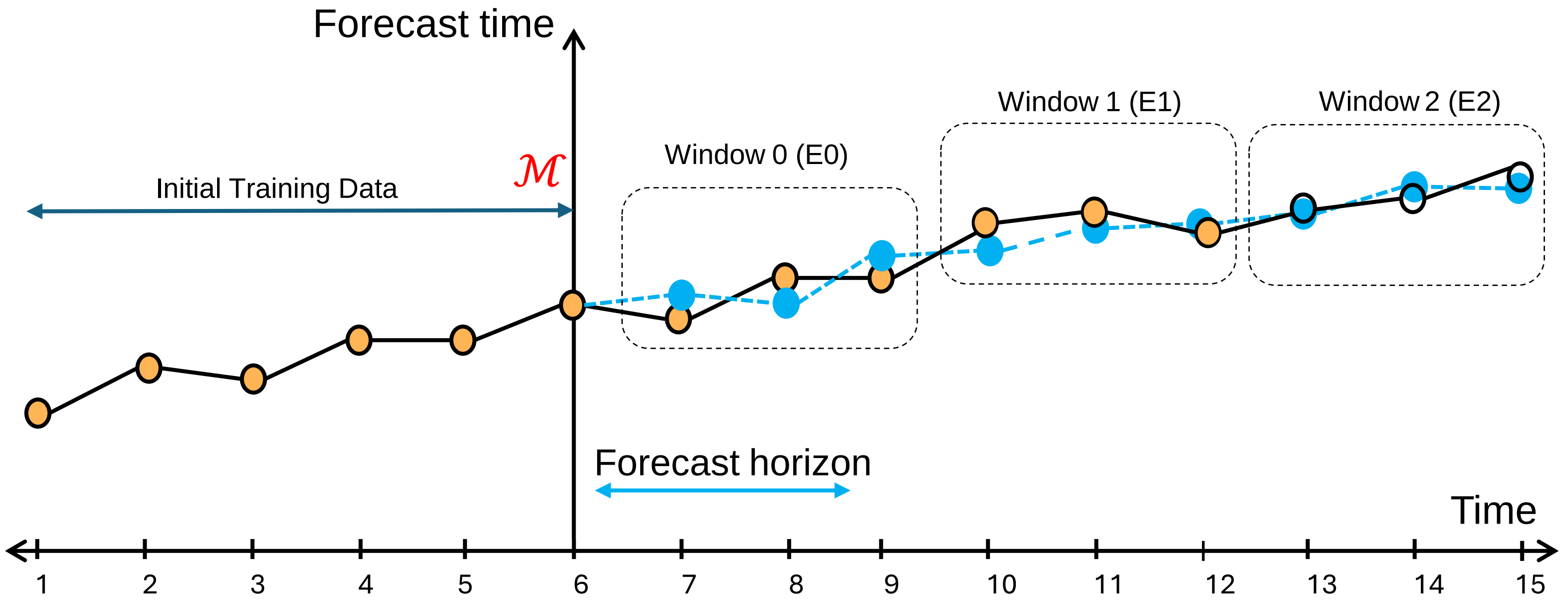
Detrending Using Differentiation

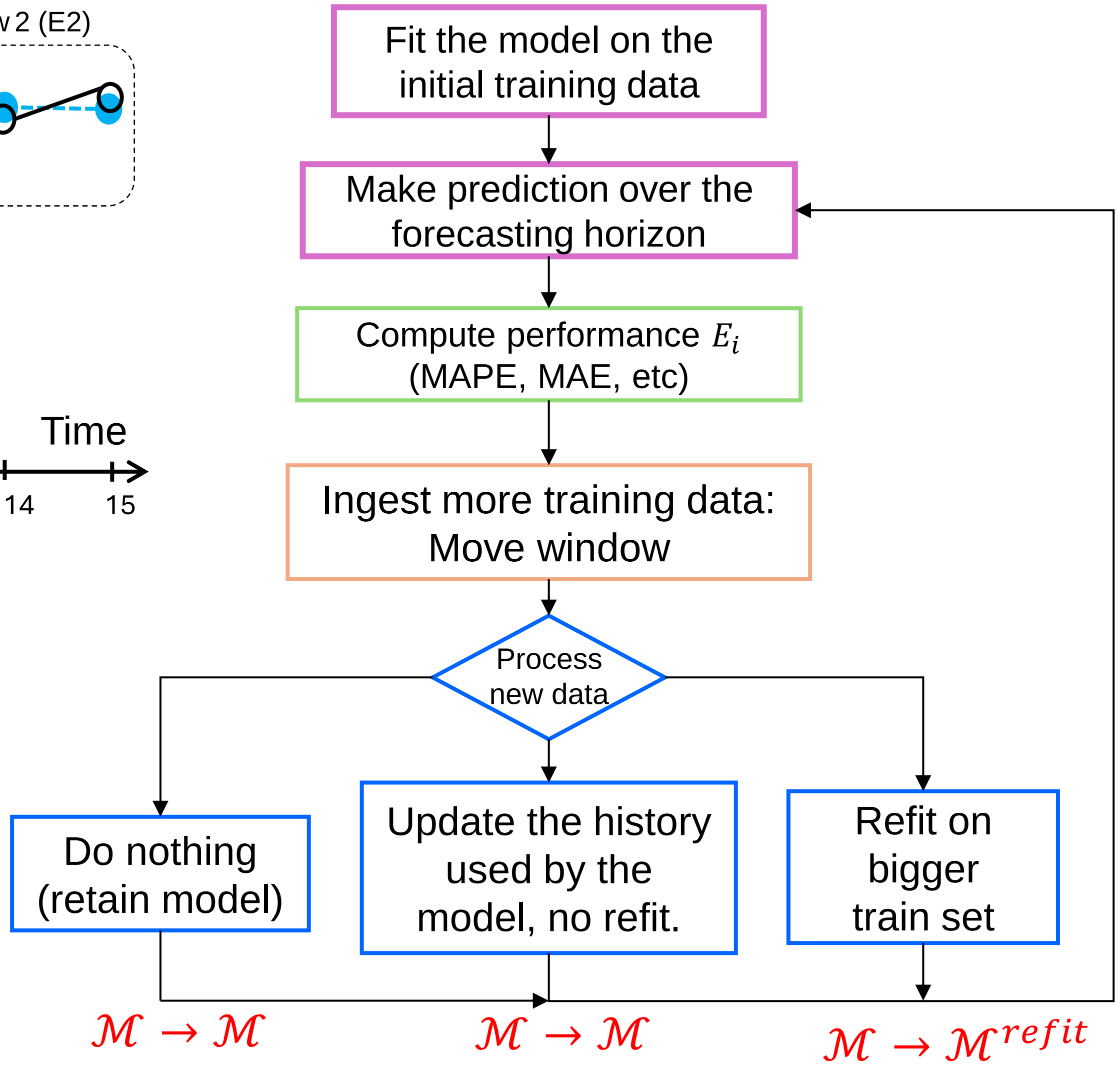
```
ts_diff = ts_no_outliers.diff().dropna()
```

Handle missing values

```
data_interpolated = data.interpolate(method='linear')
```

Backtesting: Forecasting libraries help scale experimentation





Forecasting libraries offer

- wide variety of modeling choices by interfacing/ implementing popular algorithms (sci-kit, XGBoost, etc)
- Standardized implementation of forecasting metrics
- Windowing functions to replicate forecasting scenarios: sliding, expanding, etc
- Model update/refit capabilities to replicate forecasting scenarios.

Backtesting: Setting Up Your Experiment Parameters

SKTIME

skforecast

Modeling Choices

```python
regressor =
        RandomForestRegressor(n_estimators=250,
        max_depth=10, random_state=123)

forecaster = make_reduction(regressor,
        window_length=6, strategy='recursive')
```

```python
regressor =
        RandomForestRegressor(n_estimators=250,
        max_depth=10, random_state=123)

        forecaster = ForecasterAutoreg(
                regressor = regressor,
                lags = 6)
```

Sliding/Expanding Window Features

```python
cv = ExpandingWindowSplitter(initial_window=240,
                             step_length=24,
                             fh=np.arange(1, 25))

results = evaluate(forecaster=forecaster,
                   y=y_train,
                   cv=cv,
                   return_data=True,
                   strategy='refit',
                   scoring=mean_absolute_percentage_error)
```

Refit/Update Features

Performance Metrics

```python
metric, predictions_backtest= backtesting_forecaster(
        forecaster = forecaster,
        y          = y_train["DUQ_MW"],
        initial_train_size = 240,
        fixed_train_size   = False,
        steps              = 24,
        metric= mean_absolute_percentage_error,
        refit              = True,
        verbose            = True,
        show_progress      = True)
```

Backtesting Experiment

Training set, Testing set and Predictions for each window

	test_MeanAbsolutePercentageError	fit_time	pred_time	len_train_window	cutoff	y_train	y_test	y_pred
0	0.030418	0.527055	0.110967	240	2006-01-10 23:00:00	DUQ_MW 2006-01-01 00:00:0...	DUQ_MW 2006-01-11 00:00:0...	DUQ_MW 2006-01-11 00...
1	0.053364	0.620258	0.102912	264	2006-01-11 23:00:00	DUQ_MW 2006-01-01 00:00:0...	DUQ_MW 2006-01-12 00:00:0...	DUQ_MW 2006-01-12 00...
2	0.038886	0.535657	0.099865	288	2006-01-12 23:00:00	DUQ_MW 2006-01-01 00:00:0...	DUQ_MW 2006-01-13 00:00:0...	DUQ_MW 2006-01-13 00...
3	0.067248	0.590521	0.106738	312	2006-01-13 23:00:00	DUQ_MW 2006-01-01 00:00:0...	DUQ_MW 2006-01-14 00:00:0...	DUQ_MW 2006-01-14 00...
4	0.063785	0.677798	0.099203	336	2006-01-14 23:00:00	DUQ_MW 2006-01-01 00:00:0...	DUQ_MW 2006-01-15 00:00:0...	DUQ_MW 2006-01-15 00...
...	...	...	...	...	...	...	...	...
85	0.027762	3.444988	0.113939	2280	2006-04-05 23:00:00	DUQ_MW 2006-01-01 00:00:0...	DUQ_MW 2006-04-06 00:00:0...	DUQ_MW 2006-04-06 00...
86	0.072161	3.599273	0.110180	2304	2006-04-06 23:00:00	DUQ_MW 2006-01-01 00:00:0...	DUQ_MW 2006-04-07 00:00:0...	DUQ_MW 2006-04-07 00...
87	0.207004	3.467539	0.110451	2328	2006-04-07 23:00:00	DUQ_MW 2006-01-01 00:00:0...	DUQ_MW 2006-04-08 00:00:0...	DUQ_MW 2006-04-08 00...
88	0.204000	3.575651	0.104502	2352	2006-04-08 23:00:00	DUQ_MW 2006-01-01 00:00:0...	DUQ_MW 2006-04-09 00:00:0...	DUQ_MW 2006-04-09 00...
89	0.086414	3.566354	0.097486	2376	2006-04-09 23:00:00	DUQ_MW 2006-01-01 00:00:0...	DUQ_MW 2006-04-10 00:00:0...	DUQ_MW 2006-04-10 00...

90 rows × 8 columns

End of each training window

Length of each window

Prediction time for each window

Training Time for each window

Error for each window

Backtesting Experiment

skforecast

`backtesting_forecaster()` -> returns all the predicted values and the overall performance metric value

```
Information of backtesting process
----------------------------------
Number of observations used for initial training: 240
Number of observations used for backtesting: 2160
    Number of folds: 90
    Number skipped folds: 0
    Number of steps per fold: 24
    Number of steps to exclude from the end of each train set before test (gap): 0

Fold: 0
    Training:   2006-01-01 00:00:00 -- 2006-01-10 23:00:00  (n=240)
    Validation: 2006-01-11 00:00:00 -- 2006-01-11 23:00:00  (n=24)
Fold: 1
    Training:   2006-01-01 00:00:00 -- 2006-01-11 23:00:00  (n=264)
    Validation: 2006-01-12 00:00:00 -- 2006-01-12 23:00:00  (n=24)
Fold: 2
    Training:   2006-01-01 00:00:00 -- 2006-01-12 23:00:00  (n=288)
    Validation: 2006-01-13 00:00:00 -- 2006-01-13 23:00:00  (n=24)
Fold: 3
    Training:   2006-01-01 00:00:00 -- 2006-01-13 23:00:00  (n=312)
    Validation: 2006-01-14 00:00:00 -- 2006-01-14 23:00:00  (n=24)
Fold: 4
    Training:   2006-01-01 00:00:00 -- 2006-01-14 23:00:00  (n=336)
    Validation: 2006-01-15 00:00:00 -- 2006-01-15 23:00:00  (n=24)
Fold: 5
...
Fold: 89
    Training:   2006-01-01 00:00:00 -- 2006-04-09 23:00:00  (n=2376)
    Validation: 2006-04-10 00:00:00 -- 2006-04-10 23:00:00  (n=24)
```

Training set, Testing set and the size for each fold can be seen when Verbosity is set to True

```
predictions_backtest
✓ 0.0s
```

	pred
2006-01-11 00:00:00	1520.168000
2006-01-11 01:00:00	1449.010000
2006-01-11 02:00:00	1398.115600
2006-01-11 03:00:00	1380.080162
2006-01-11 04:00:00	1355.504533
...	...
2006-04-10 19:00:00	1690.002351
2006-04-10 20:00:00	1676.797505
2006-04-10 21:00:00	1671.707992
2006-04-10 22:00:00	1678.172647
2006-04-10 23:00:00	1682.843514

2160 rows × 1 columns

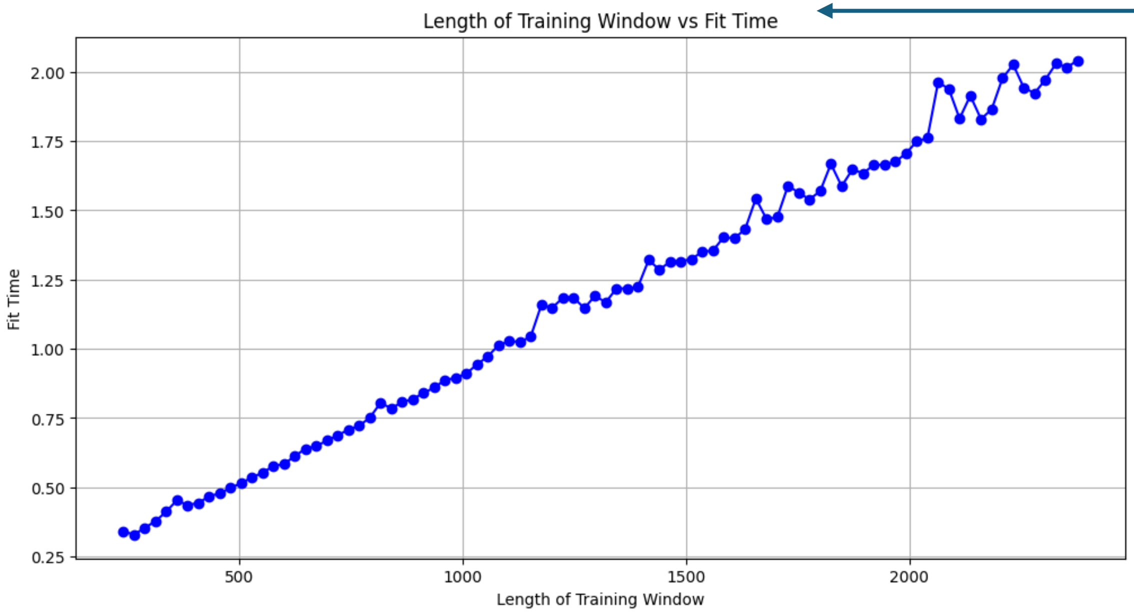
```
metric
✓ 0.0s
```

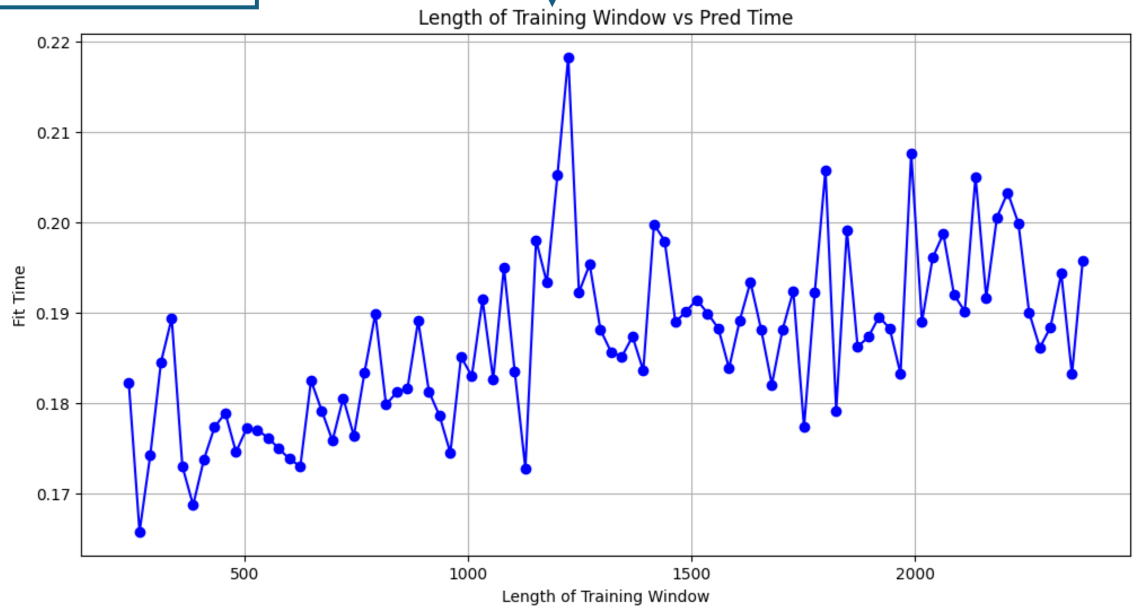
	mean_absolute_percentage_error
0	0.062876

Backtesting Experiment

Backtesting Parameters:
window_movement = expanding
initial_data = 240
fh = 24
window_stride = 24
metric = MAPE
refit_update = refit
window_length = 24

	SKTIME	skforecast
Experiment run time (ms)	117357.61	33883.11
Experiment Metric (MAPE)	0.088353	0.062876
Window length vs Training time		-
Window Length vs Prediction time		-



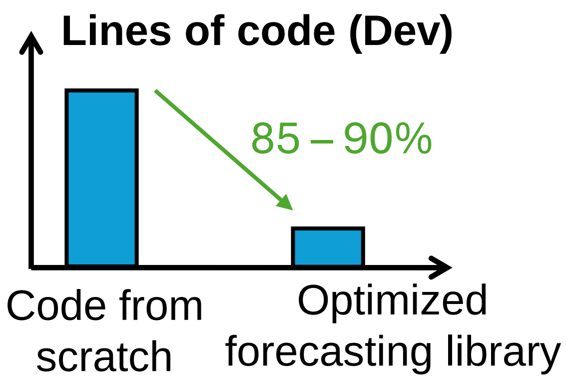


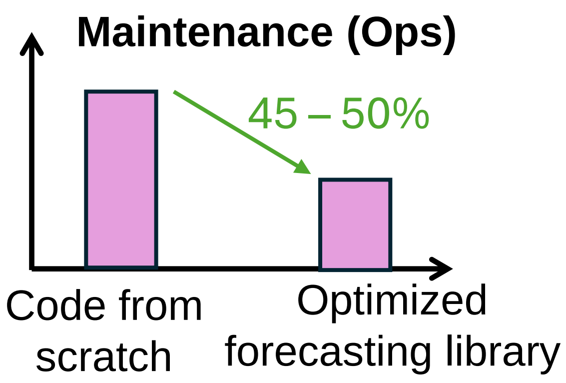
Conclusion and Key Takeaways

Minimize boilerplate code accelerate development & simplify maintenance

SKTIME skforecast

- **Data Understanding**
- **Data Preparation**
- **Backtesting**





- SKTime and SKForecast: two popular forecasting libraries. SKForecast focuses on a core set of features; SKTime is all encompassing.
- **Data Understanding**: SKTime hooks into `statsmodels` to provide several visualization functions.
- **Data Preparation**: SKTime provides several utilities, whereas SKForecast expects the data scientist to use Pandas preprocessing functions.
- **Backtesting**: Despite limited functionality, SKForecast is quite fast. SKTime provides more options and fine-grained stats.