Non-Sequential Machine Learning Pipelines with pyWATTS

deRSE23 - Conference for Research Software Engineering in Germany

Benedikt Heidrich, Kaleb Phipps,
Stefan Meisenbacher, Marian Turowski, Oliver Neumann, Ralf Mikut, Veit Hagenmeyer
Machine Learning Pipelines

Input

Pre-Processing

Task (Forecasting, Classification)

Output
Machine Learning Pipelines

Input → Pre-Processing → Task (Forecasting, Classification) → Output
Machine Learning Pipelines

Pipeline

Input → Pre-Processing → Task (Forecasting, Classification) → Output
Machine Learning Pipelines

- Why is this beneficial?
  - Hyperparameter tuning is simpler - you can tune the whole pipeline.
  - It is easy to handle - there is only one pipeline object that needs to be saved.
  - You only have to call the fit method once to train the entire pipeline.
Machine Learning Pipelines

- Why is this beneficial?
  - Hyperparameter tuning is simpler - you can tune the whole pipeline.
  - It is easy to handle - there is only one pipeline object that needs to be saved.
  - You only have to call the fit method once to train the entire pipeline.
- Existing sequential pipeline implementations in...
  - sklearn
  - sktime
However: Many use cases are non-sequential

For example, electricity price forecasting

- **Price**
  - Standard Scaler
  - Price Forecaster
  - Inverse Scaling

- **Time Series Index**
  - Calendar Features
  - Standard Scaler
  - Inverse Scaling

- **Load**
  - Standard Scaler
  - Load Forecaster
  - Inverse Scaling
pyWATTS: Python Workflow Automation Tool for Time Series

- pyWATTS models pipeline as directed acyclic graphs enabling non-sequential workflows:
  - Code is easier to write and intuitive to understand.
  - Hyperparameter optimisation is easier.
  - Possible to combine multiple tasks in one pipeline.

- pyWATTS provides three different APIs for creating pipelines:
  - Functional API
  - Imperative API
  - Constructor API
pyWATTS: Python Workflow Automation Tool for Time Series

- pyWATTS models pipeline as directed acyclic graphs enabling non-sequential workflows:
  - Code is easier to write and intuitive to understand.
  - Hyperparameter optimisation is easier.
  - Possible to combine multiple tasks in one pipeline.

- pyWATTS provides three different APIs for creating pipelines:
  - Functional API
  - Imperative API
  - Constructor API

For Today: Focus on the Functional API
Example of pyWATTS Code

```python
pipeline = Pipeline(path="../results")

calendar = CalendarExtraction(continent="Europe", country="Germany", features=[CalendarFeature.month, CalendarFeature.weekday, CalendarFeature.weekend], name="calendar")
(x=pipeline["load_power_statistics"])

[... # Extract Lag Features]

forecast_load = SKLearnWrapper(module=LinearRegression(fit_intercept=True), name="load_forecast")(features=lag_features_load, calendar=calendar, target=target)

forecast_price_scaled = SKLearnWrapper(module=LinearRegression(fit_intercept=True), name="price_forecast")(features=lag_features_price, calendar=calendar, load=forecast_load, target=target_price)

[... # Evaluate Forecast]

pipeline.train(data)
pipeline.test(data)
```
Example of pyWATTS Code

```python
pipeline = Pipeline(path="../results")

calendar = CalendarExtraction(continent="Europe",
                               country="Germany",
                               features=[CalendarFeature.month, CalendarFeature.weekday,
                                         CalendarFeature.weekend],
                               name="calendar")

(x=pipeline["load_power_statistics"])

[... # Extract Lag Features]

forecast_load = SKLearnWrapper(module=LinearRegression(fit_intercept=True), name="load_forecast")
(features=lag_features_load, calendar=calendar, target=target)

forecast_price_scaled = SKLearnWrapper(module=LinearRegression(fit_intercept=True),
                                        name="price_forecast")
(features=lag_features_price, calendar=calendar, load=forecast_load, target=target_price)

[... # Evaluate Forecast]

pipeline.train(data)
pipeline.test(data)
```
Example of pyWATTS Code

```python
calendar = CalendarExtraction(continent="Europe",
                               country="Germany",
                               features=[CalendarFeature.month, CalendarFeature.weekday,
                                         CalendarFeature.weekend],
                               name="calendar")
(x=pipeline["load_power_statistics"])
```

```python
forecast_load = SKLearnWrapper(module=LinearRegression(fit_intercept=True), name="load_forecast")
(features=lag_features_load, calendar=calendar, target=target)
```

```python
forecast_price_scaled = SKLearnWrapper(module=LinearRegression(fit_intercept=True),
                                        name="price_forecast")
(features=lag_features_price, calendar=calendar, load=forecast_load, target=target_price)
```

```python
pipeline.train(data)
pipeline.test(data)
```
Example of pyWATTS Code

```python
target_price = pipeline["load_power_statistics"]

forecast_load = SKLearnWrapper( module=LinearRegression( fit_intercept=True), name="load_forecast")
    features=lag_features_load, calendar=calendar, target=target)

forecast_price_scaled = SKLearnWrapper( module=LinearRegression( fit_intercept=True), name="price_forecast")
    features=lag_features_price, calendar=calendar, load=forecast_load, target=target_price)

pipeline.train(data)
pipeline.test(data)
```
Example of pyWATTS Code

```python
pipeline = Pipeline(path="../results")

calendar = CalendarExtraction(continent="Europe",
                               country="Germany",
                               features=[CalendarFeature.month, CalendarFeature.weekday,
                                         CalendarFeature.weekend],
                               name="calendar")
(x=pipeline["load_power_statistics"])

[... # Extract Lag Features]

forecast_load = SKLearnWrapper(module=LinearRegression(fit_intercept=True), name="load_forecast")
(features=lag_features_load, calendar=calendar, target=target)

forecast_price_scaled = SKLearnWrapper(module=LinearRegression(fit_intercept=True),
                                        name="price_forecast")
(features=lag_features_price, calendar=calendar, load=forecast_load, target=target_price)

[... # Evaluate Forecast]

pipeline.train(data)
pipeline.test(data)
```
Example of pyWATTS Code

```python
pipeline = Pipeline(path="../results")

calendar = CalendarExtraction( continent="Europe",
                                country="Germany",
                                features=[CalendarFeature.month, CalendarFeature.weekday,
                                          CalendarFeature.weekend],
                                name="calendar")
(x=pipeline["load_power_statistics"])

[... # Extract Lag Features]

forecast_load = SKLearnWrapper(module=LinearRegression(fit_intercept=True), name="load_forecast")
(features=lag_features_load, calendar=calendar, target=target)

forecast_price_scaled = SKLearnWrapper(module=LinearRegression(fit_intercept=True),
                                       name="price_forecast")
(features=lag_features_price, calendar=calendar, load=forecast_load, target=target_price)

[... # Evaluate Forecast]

pipeline.train(data)
pipeline.test(data)
```
Visualisation of resulting pyWATTS Pipeline

drawing = pipeline.draw()
drawing.draw()
Easy Hyperparameter Tuning

```python
params = {
    "load_forecast__module": [LinearRegression(), MLPRegressor()],
    "price_forecast__module": [LinearRegression(), MLPRegressor()],
    "scaler_power__module": [MinMaxScaler(), StandardScaler()],
    ...
}
```
Easy Hyperparameter Tuning

```python
params = {
    "load_forecast__module": [LinearRegression(), MLPRegressor()],
    "price_forecast__module": [LinearRegression(), MLPRegressor()],
    "scaler_power__module": [MinMaxScaler(), StandardScaler()],
    ...
}

from sklearn.model_selection import TimeSeriesSplit
tscv = TimeSeriesSplit(test_size=168*4)
pipeline_cv = GridSearchCV(pipeline, param_grid=params, cv=tscv)
pipeline_cv.fit(data)
```
Easy Hyperparameter Tuning

define hyperparameter grid
params = {
    "load_forecast__module": [LinearRegression(), MLPRegressor()],
    "price_forecast__module": [LinearRegression(), MLPRegressor()],
    "scaler_power__module": [MinMaxScaler(), StandardScaler()],
    ...
}

from sklearn.model_selection import TimeSeriesSplit

tscv = TimeSeriesSplit(test_size=168*4)

pipeline_cv = GridSearchCV(pipeline, param_grid=params, cv=tscv)

pipeline_cv.fit(data)

pipeline_cv.best_params_
Easy Hyperparameter Tuning

```python
params = {
    "load_forecast__module": [LinearRegression(), MLPRegressor()],
    "price_forecast__module": [LinearRegression(), MLPRegressor()],
    "scaler_power__module": [MinMaxScaler(), StandardScaler()],
    ...
}

from sklearn.model_selection import TimeSeriesSplit
tscv = TimeSeriesSplit(test_size=168*4)
pipeline_cv = GridSearchCV(pipeline, param_grid=params, cv=tscv)
pipeline_cv.fit(data)

pipeline_cv.best_params_
```

Out[14]:

```python
dict({'calendar__features': [<CalendarFeature.month_cos: 4>,
    <CalendarFeature.month_sin: 3>,
    <CalendarFeature.weekend: 21>],
    'load_forecast__module': LinearRegression(),
    'price_forecast__module': LinearRegression(),
    'scaler_power__module': MinMaxScaler(),
    'scaler_price_module': MinMaxScaler()})
```
Easy Hyperparameter Tuning

```python
params = {
    "load_forecast__module": [LinearRegression(), MLPRegressor()],
    "price_forecast__module": [LinearRegression(), MLPRegressor()],
    "scaler_power__module": [MinMaxScaler(), StandardScaler()],
    ...
}

from sklearn.model_selection import TimeSeriesSplit
tscv = TimeSeriesSplit(test_size=168*4)
pipeline_cv = GridSearchCV(pipeline, param_grid=params, cv=tscv)
pipeline_cv.fit(data)

pipeline_cv.best_params_
```

```
Out[14]:
{'calendar__features': [<CalendarFeature.month_cos: 4>],
 'calendar__features': [<CalendarFeature.month_sine: 3>],
 'calendar__features': [<CalendarFeature.weekend: 21>],
 'load_forecast__module': LinearRegression(),
 'price_forecast__module': LinearRegression(),
 'scaler_power__module': MinMaxScaler(),
 'scaler_price__module': MinMaxScaler()}
```
Use-Cases Realised With pyWATTS

- Enhancing anomaly detection methods (see our poster for more information):
  - We use latent space data representations of energy time series to improve the performance of anomaly detection methods.
  - Read our anomaly detection paper here!

- (Probabilistic) Profile Neural Network: (Prob)PNN (see our poster for more information):
  - We incorporate statistical information in the form of profiles into deep learning methods to improve (probabilistic) time series forecasting.
  - Read the paper on the deterministic PNN here and the probabilistic PNN here!

- AutoPV: Automated Photovoltaic Forecasts:
  - We use information about PV configurations to create a cold-start capable ensemble of PV power forecasting models.
  - Read our AutoPV paper here!

- Creating Probabilistic Forecasts from arbitrary deterministic forecasts:
  - We use a conditional invertible neural network to transform an arbitrary deterministic forecast into a probabilistic forecast.
  - Read our paper on creating probabilistic forecasts here!
Development Roadmap & How to Contribute

- We aim to grow and develop pyWATTS in the coming months by:
  - Integrating further with sktime
  - Increasing performance by enabling parallel execution of pipeline components.
Development Roadmap & How to Contribute

- We aim to grow and develop pyWATTS in the coming months by:
  - Integrating further with `sktime`
  - Increasing performance by enabling parallel execution of pipeline components.

- We are looking to grow the pyWATTS community - so if you are interested in contributing please get in touch!
  - Join the pyWATTS community as a user or developer:
    - GitHub - pyWATTS: Python Workflow Automation Tool for Time-Series
    - GitHub - pywatts-pipeline
  - Contact us if you want more information:
    - pywatts-team@iai.kit.edu
Thank you for your Attention

pyWATTS-pipeline

https://github.com/KIT-IAI/pywatts-pipeline

deRSE Jupyter Notebooks

https://github.com/KIT-IAI/pyWATTS-deRSE-2023

Benedikt Heidrich: LinkedIn: benedikt-heidrich; benedikt.heidrich@kit.edu

Kaleb Phipps: LinkedIn: kaleb-phipps; kaleb.phipps@kit.edu