AutoGluon.TimeSeries 0.7.0

Installation

AutoGluon (GitHub) requires pip > 1.4 (upgrade by pip install -U pip). More installation options. AutoGluon supports Python 3.8 to 3.10. Installation is available for Linux, MacOS, and Windows.

```
pip install autogluon
```

Preparing Data

AutoGluon.TimeSeries accepts datasets with multiple univariate time series. Here we use the M4 Competition Daily dataset to demonstrate how to do forecasting with AutoGluon.TimeSeries.

```
import pandas as pd
raw_data = pd.read_csv("m4_daily.csv")
raw_data.head()
```

	item_id	timestamp	target	weekend
0	D3937	1989-03-03 12:00:00	4500.0	0.0
1	D3937	1989-03-04 12:00:00	4450.0	1.0
2	D3937	1989-03-05 12:00:00	4450.0	1.0

Each row contains unique ID of each time series, timestamp, value of the time series, and (optionally) time-varying **covariates**.

Convert raw data into a **TimeSeriesDataFrame** used by AutoGluon.

```
from autogluon.timeseries import TimeSeriesDataFrame
train_data = TimeSeriesDataFrame.from_data_frame(
    raw_data,
    id_column="item_id",
    timestamp_column="timestamp",
)
```

TimeSeriesDataFrame can also store time-independent **static features** (metadata) for each time series.

```
raw_static_features = pd.read_csv(
    "m4_metadata.csv", index_col=0
)
raw_static_features.head()
```

	category
item_id	
D3937	Other
D1897	Industry
D2249	Finance
D1580	Micro

```
train_data.static_features = raw_static_features
```

Training

Train models to forecast the values in the column 'target' 30 steps into the future for each time series.

```
from autogluon.timeseries import TimeSeriesPredictor
predictor = TimeSeriesPredictor(
    target="target",
    prediction_length=30,
).fit(train_data)
```

More options to construct a **TimeSeriesPredictor** instance (docs):

```
# The metric used to tune models
eval_metric="MAPE"
# Select quantiles for the probabilistic forecast
quantile_levels = [0.1, 0.5, 0.9]
# Covariates that are known in the future
# (e.g., holidays, promotions, weather forecasts)
known_covariates_names=["weekend"]
# Evaluate models with multi-window backtesting
validation_splitter="multi_window"
# Train on irregular time series
ignore_time_index=True
```

More options for the **fit** method (docs):

```
# Limit the training time, in second
time_limit=600
# Train more models for more accurate forecasts,
# but longer training time.
presets="high_quality"
# Use a separate dataset to tune models.
tuning_data=val_data
# Manually select what models to train.
# E.g., only train ETS with seasonal_period=14
# and DeepAR with default hyperparameters
hyperparameters={
    "ETS": {"seasonal_period": 14},
    "DeepAR": {},
}
```

Monitoring

Understand the contribution of each model.

```
predictor.leaderboard()
```

Validation score
Inference time
Training time
The model ensembles all
Individual model

	model	score_val	pred_time_val	fit_time_marginal
0	WeightedEnsemble	-0.964	92.354	177.439
1	AutoGluonTabular	-1.018	3.127	25.023
2	SeasonalNaive	-1.081	1.076	0.001
3	DeepAR	-1.013	13.004	512.672
4	ETS	-1.595	56.118	0.001

Predicting

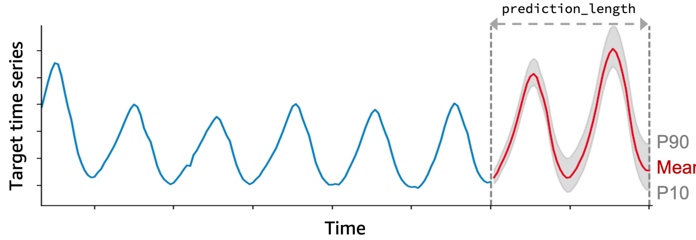
Forecast `prediction_length` steps into the future starting from the end of each time series in `train_data`.

```
predictions = predictor.predict(
    train_data,
    # only necessary if known_covariates_names
    # were provided when creating predictor
    known_covariates=known_covariates,
)
known_covariates.head()
```

		weekend
item_id	timestamp	
D3937	1989-12-01 12:00:00	0.0
	1989-12-02 12:00:00	1.0
	1989-12-03 12:00:00	1.0

AutoGluon generated probabilistic forecasts that include

- mean forecast — expected value of the time series
- quantile forecast — range of possible outcomes




```
predictions.head()
```

		mean	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
item_id	timestamp										
D3937	1989-12-01 12:00:00	5282.95	5229.38	5247.77	5261.03	5272.36	5282.95	5293.53	5304.86	5318.12	5336.51
	1989-12-02 12:00:00	5285.85	5210.61	5236.44	5255.06	5270.98	5285.85	5300.72	5316.64	5335.26	5361.09
	1989-12-03 12:00:00	5288.74	5196.98	5228.48	5251.19	5270.60	5288.74	5306.88	5326.28	5349.00	5380.49

AutoGluon predicts with the final ensemble model. You can also predict using an individual model.

```
models = predictor.get_model_names()
predictor.predict(test_data, model=models[1])
```

- Detailed time series tutorials.
- For other types of data, check TabularPredictor for tabular data and MultiModalPredictor for multi-modal data such as images and text.
- Check the latest version of this cheat sheet.
- Any questions? Ask here
- Like what you see? Consider starring AutoGluon on GitHub and following us on Twitter to get notified of the latest updates!