Data Science Seminar: Time Series Forecasting with Tree-Based Ensemble Methods

February 25, 2022

About the Presenter

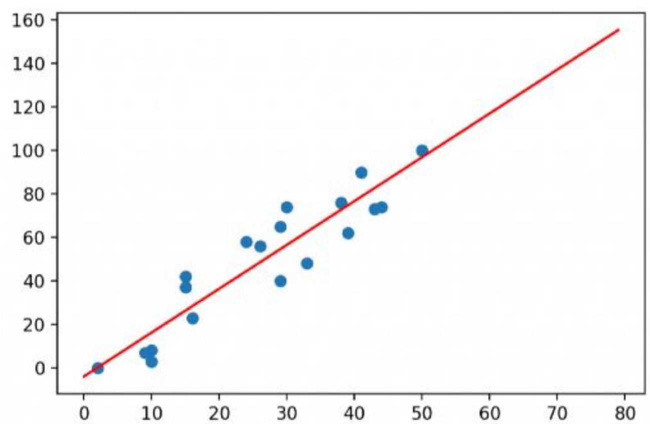
- Brett Efaw
- Data Scientist at Idaho Power Company
- BEfaw@idahopower.com or brettefaw@gmail.com
- Work with various business groups to implement Data Science solutions
  - Decision support
  - Predictive modeling
  - Forecasting
  - BI and DS reporting/analysis applications

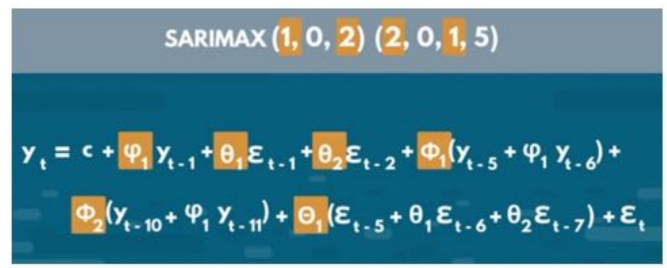
Agenda

- Traditional vs. (non-DL) ML Forecasting Methods
- Simulated daily data set in Python
- Trend + Tree-based ensemble
  - Feature engineering from date components
  - De-trend (fit past and forecast future trend)
  - Train tree-based model on de-trended target ~ date components
  - Combine trend forecast + tree-based prediction
- Compare to traditional time series method (ARIMA)
- Discuss pros and cons of this approach

Traditional Forecasting Methods

- Predict future values as function of past values
  - Classic statistical methods: linear regression (or similar) to extrapolate (trend) into the future
  - Classic time series methods: auto-regressive = regress data onto itself (from past) into the future





Forecasting with Machine Learning

- Predict future values as function of (calendar) features
  - Extract features from Date/Time:
    - Day of week, month, quarter, year
    - Month of year
    - Quarter of year
    - Hour of day, week, month, year
    - Minute of hour, day
    - AM/PM
    - Holiday/special event
    - Business-specific process/change
  - Some features are cyclic and can be transformed to a trigonometric scale
  - Take advantage of knowing these features for future dates/times

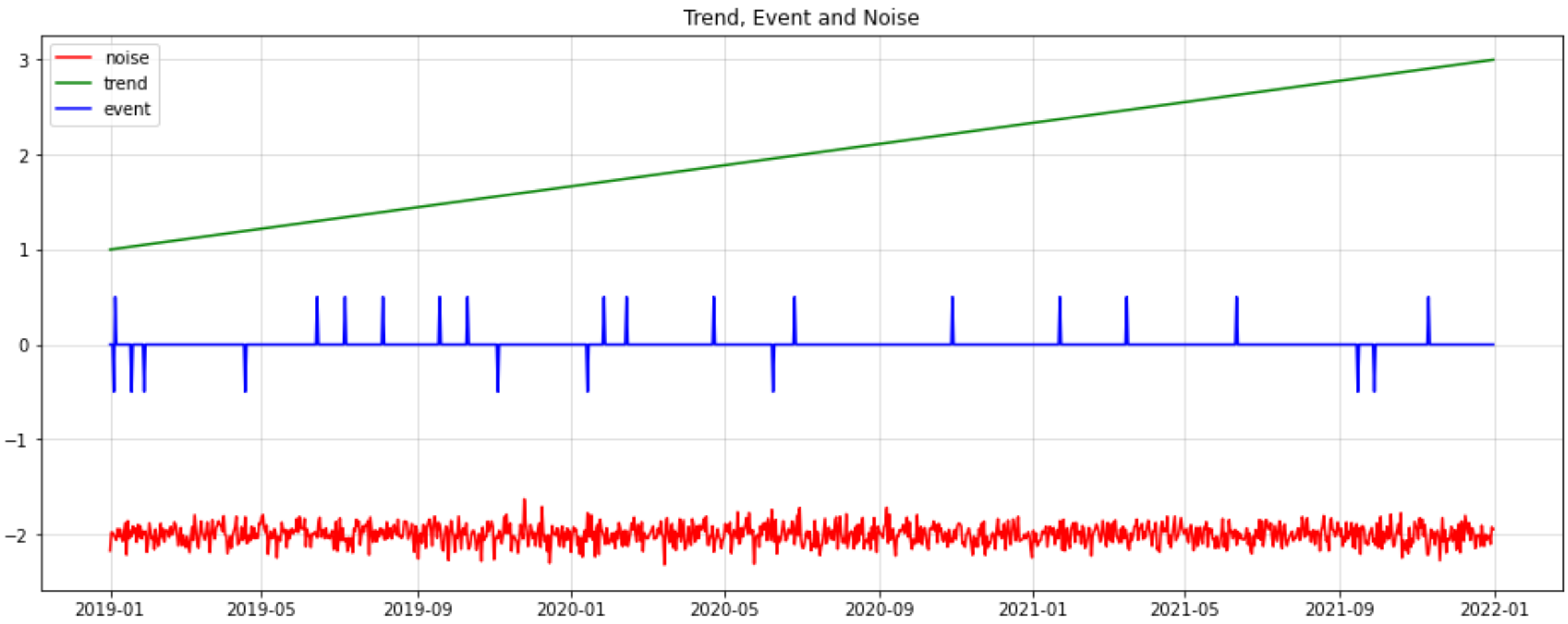
Date	DOW	DOM	DOY	MOY	QTR
2/3/2022	5	3	332	2	1
2/4/2022	6	4	331	2	1
2/5/2022	7	5	330	2	1
2/6/2022	1	6	329	2	1
2/7/2022	2	7	328	2	1
2/8/2022	3	8	327	2	1
2/9/2022	4	9	326	2	1
2/10/2022	5	10	325	2	1
2/11/2022	6	11	324	2	1
2/12/2022	7	12	323	2	1
2/13/2022	1	13	322	2	1
2/14/2022	2	14	321	2	1
2/15/2022	3	15	320	2	1
2/16/2022	4	16	319	2	1
2/17/2022	5	17	318	2	1
2/18/2022	6	18	317	2	1
2/19/2022	7	19	316	2	1
2/20/2022	1	20	315	2	1
2/21/2022	2	21	314	2	1
2/22/2022	3	22	313	2	1

Simulated Data Set

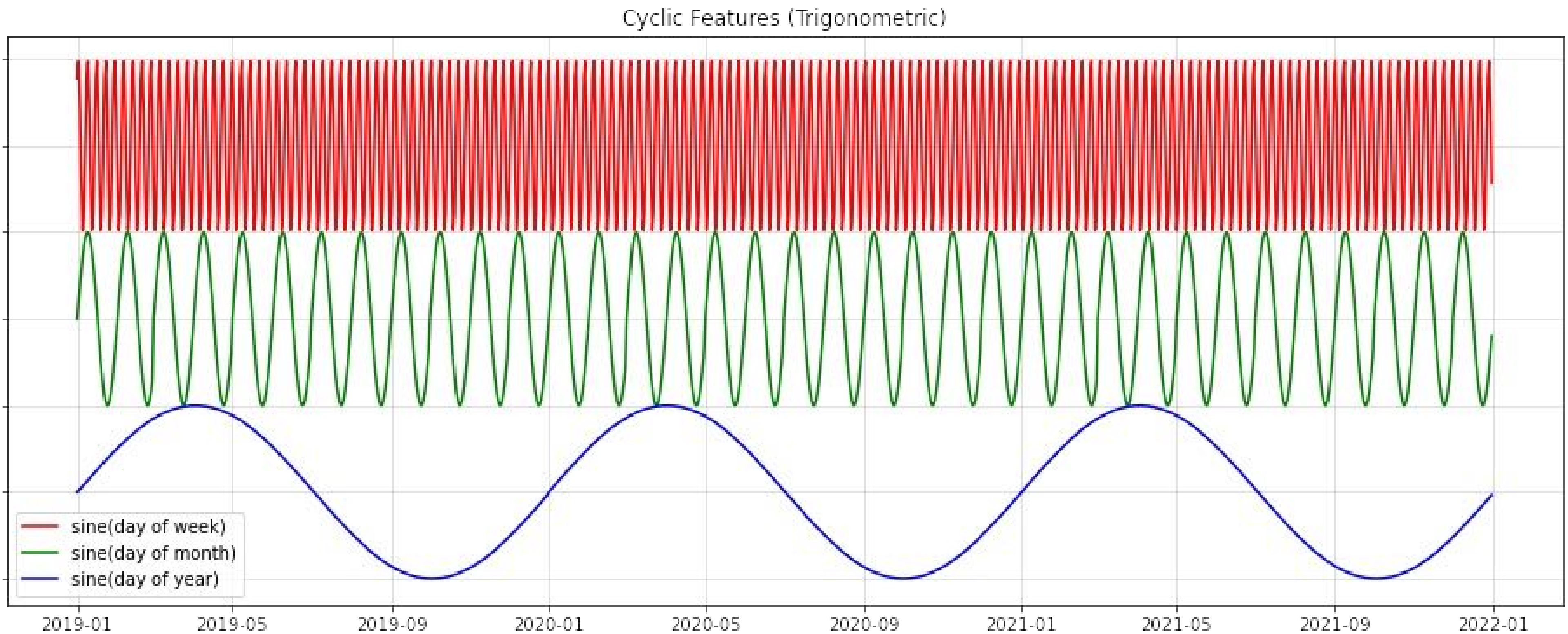
- Create a Pandas data frame with 3 years of daily data using date_range()
- Fill this data frame with columns for:
  - noise: random noise from N(0,0.1)
  - trend: linear trend from -1 to 1
  - event: -0.5 (dip), 0.5 (spike) or 0.0
  - dw: day of week
  - dm: day of month
  - dy: day of year
  - dn: day (row) index
- Trig Conversion for Date Components:
  - sdw: sine representation for day of week
  - sdm: sine representation for day of month
  - sdy: sine representation for day of year
- Target Variable:
  - trend + sine components + event + noise

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df = pd.DataFrame()
df["dt"] = pd.date_range("2019-01-01", "2021-12-31", freq = "D")
df["noise"] = np.random.normal(0, 0.1, len(df))
df["trend"] = np.linspace(-1.0, 1.0, len(df))
df["event"] = np.random.choice([-0.5, 0.0, 0.5], size = len(df), p = [0.01, 0.98, 0.01])
df["dw"] = df["dt"].dt.dayofweek+1
df["dm"] = df["dt"].dt.day
df["dy"] = df["dt"].dt.dayofyear
df["dn"] = df.index + 1
df["sdw"] = np.sin(2.0 * np.pi * (df["dw"]-1)/7)
df["sdm"] = np.sin(2.0 * np.pi * (df["dm"]-1)/31)
df["sdy"] = np.sin(2.0 * np.pi * (df["dy"]-1)/366)
df["y"] = df["trend"] + (0.333*df["sdw"]) + (0.333*df["sdm"]) + (0.333*df["sdy"]) + df["event"] + df["noise"]
```

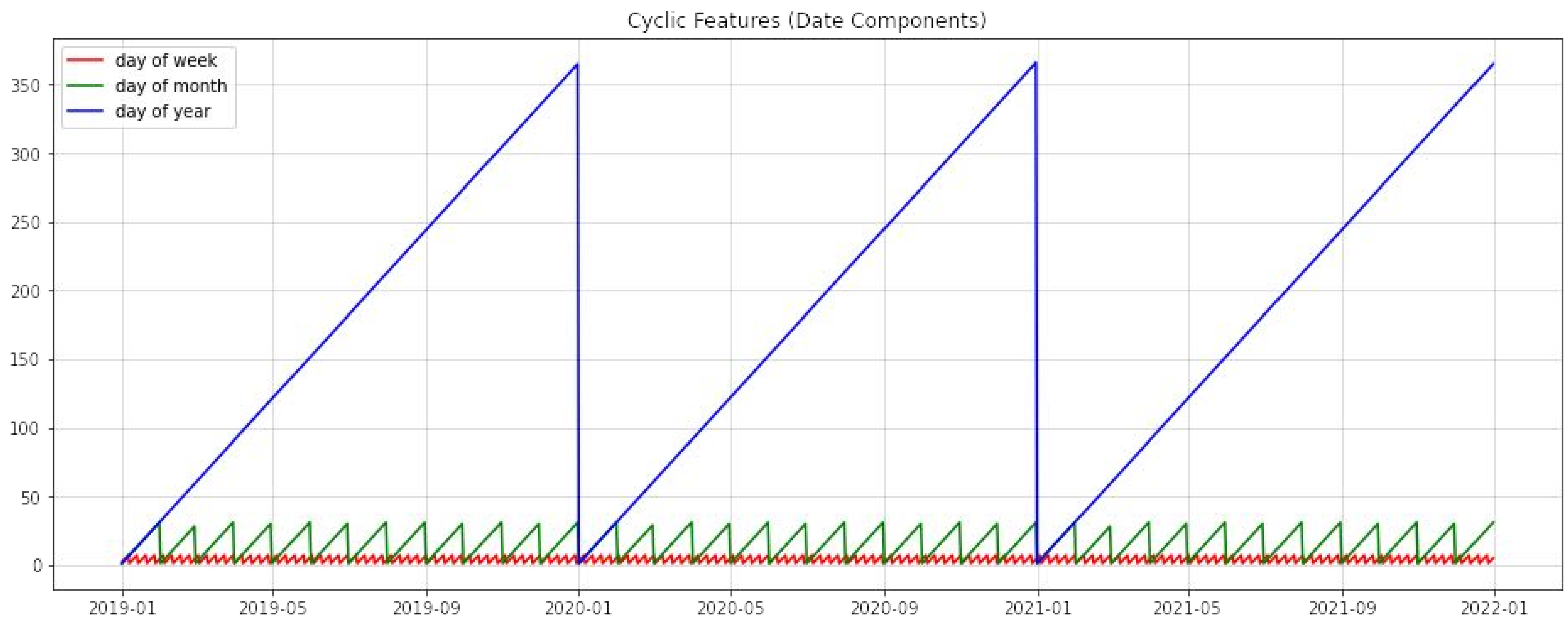
Trend, Event and Noise



Cyclic Features (Trigonometric)

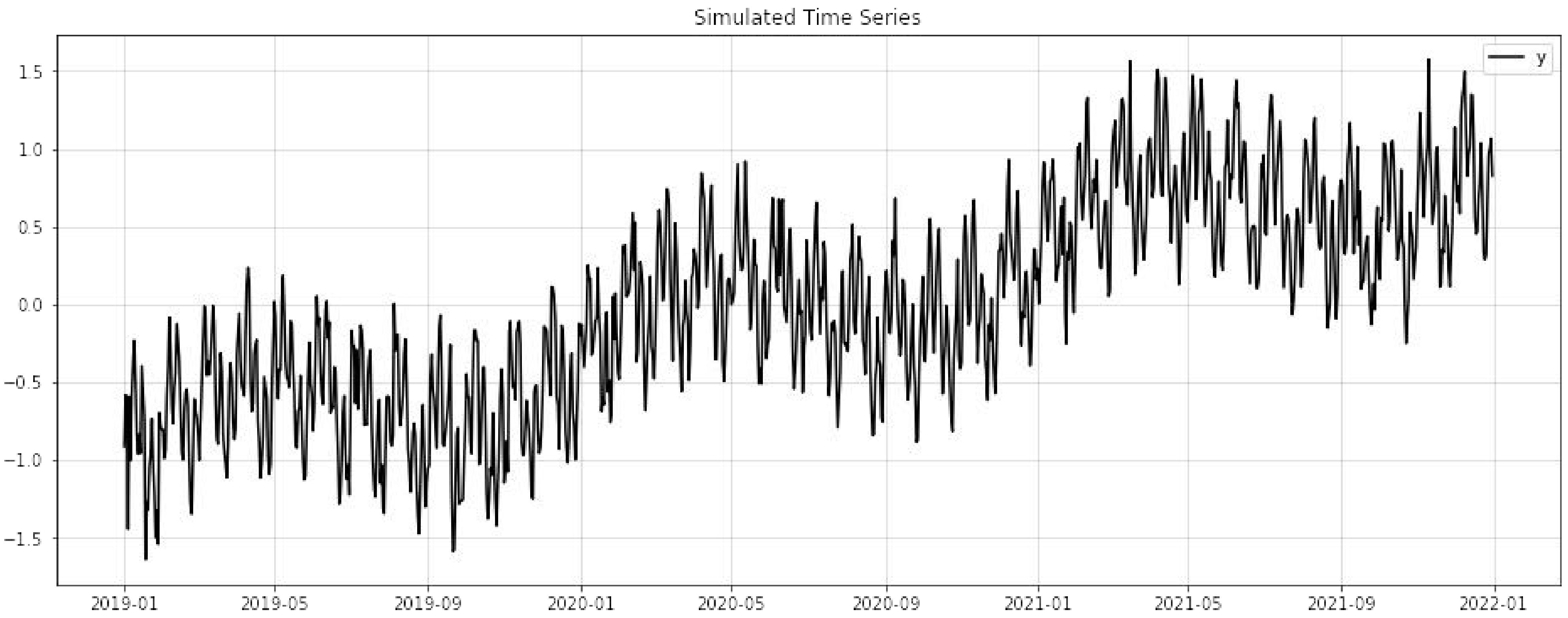


Cyclic Features (Date Components)



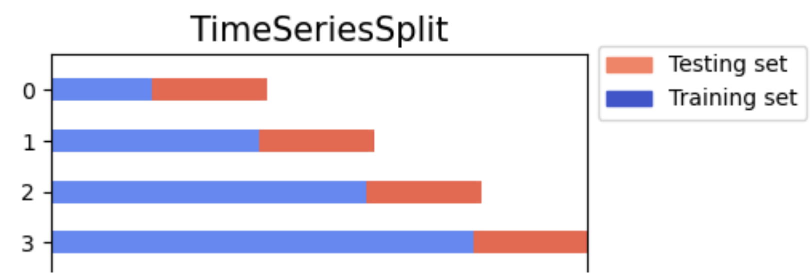
Target Variable

Long-term trend
Multiple seasonal periods



Time Series Splitting

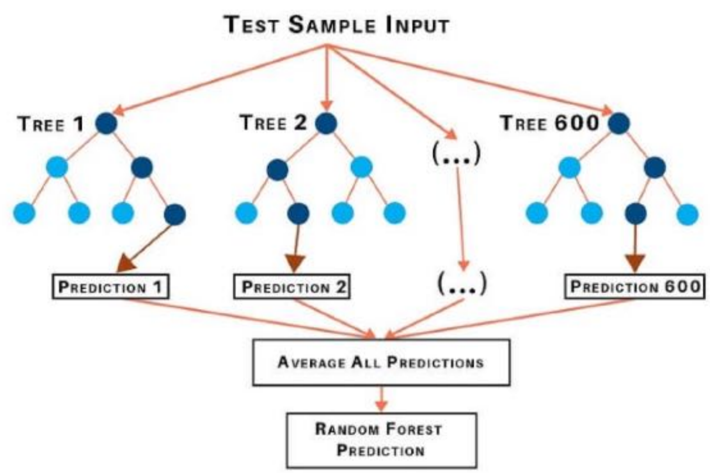
- Primarily used for evaluating a model's forecast accuracy
- Split data into training/testing partitions while preserving order
- Number of splits (like number of folds in cross-validation)
  - For example: 10 splits
- Test size = desired forecast horizon
  - For example: 30 dates in future
- Maximum train size (default is all past data before test partition)
- See TimeSeriesSplit from sklearn.model_selection for more details [sklearn.model_selection.TimeSeriesSplit — scikit-learn 1.0.2 documentation]



[Visualizing cross-validation behavior in scikit-learn — scikit-learn 1.0.2 documentation]

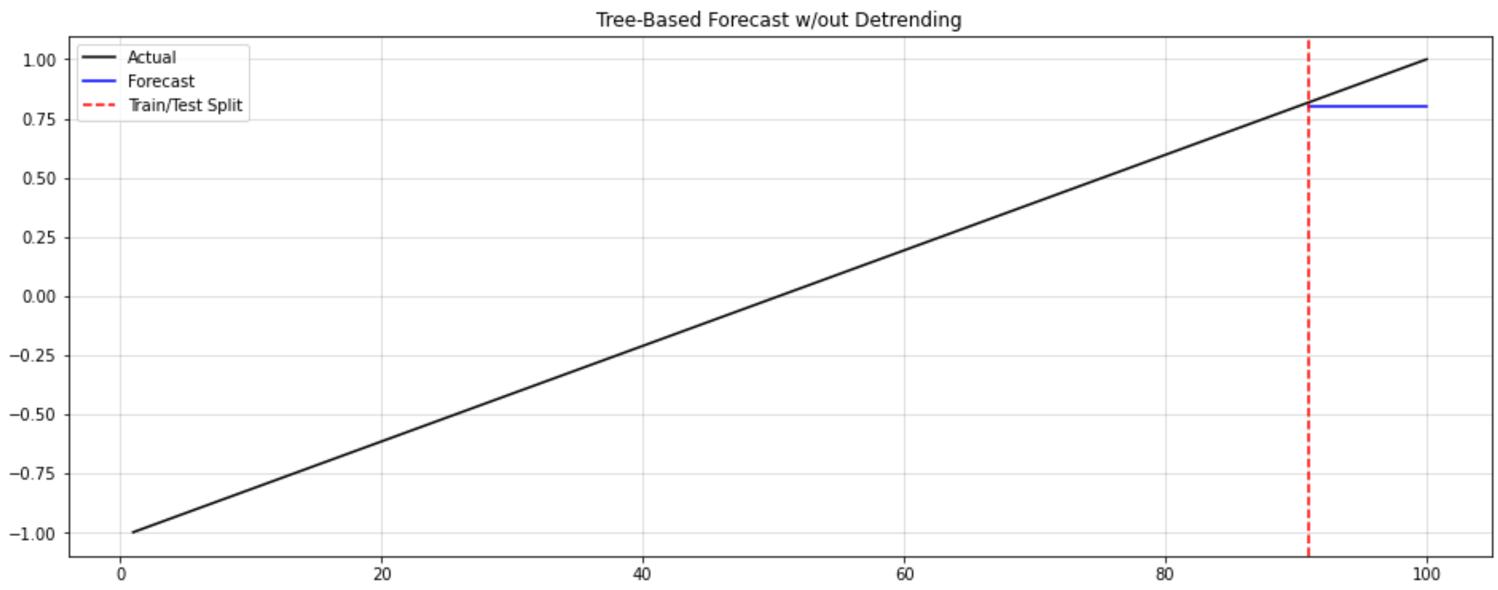
Tree-Based Ensemble Methods

- Tree-based ensembles are an extension of Decision Trees
- Very flexible and fast algos used for both classification and regression
- Can handle numeric or categorical inputs
- No feature scaling/normalization required
- Random Forests
  - [sklearn.ensemble.RandomForestRegressor — scikit-learn 1.0.2 documentation]
- Gradient Boosting
  - [sklearn.ensemble.GradientBoostingRegressor — scikit-learn 1.0.2 documentation]



Tree-Based Ensembles Can't Extrapolate

- One fundamental drawback for tree-based models:
  - Can't predict outside the range of observed values
  - This can be a problem if the target variable extrapolates outside the range of past values



Trend + Tree-Based Ensemble

- y: Target variable
- dn: day number (row sequence)
- **Trend Model Formula:**
  - **y = f(dn)**

- ydetrend: Target variable detrended
- dw: day of week (1-7)
- dm: day of month (1-31)
- dy: day of year (1-366)
- **Tree-Based Model Formula:**
  - **ydetrend = f(dw, dm, dy, event)**

dt	noise	trend	event	dw	dm	dy	dn	sdw	sdm	sdy	y
2019-01-01	0.103337	-1.000000	0.0	2	1	1	1	0.781831	0.000000	0.000000	-0.636313
2019-01-02	-0.108721	-0.998174	0.0	3	2	2	2	0.974928	0.201299	0.017166	-0.709494
2019-01-03	-0.038267	-0.996347	0.0	4	3	3	3	0.433884	0.394356	0.034328	-0.747379
2019-01-04	0.069619	-0.994521	0.0	5	4	4	4	-0.433884	0.571268	0.051479	-0.862010
2019-01-05	0.020431	-0.992694	0.0	6	5	5	5	-0.974928	0.724793	0.068615	-1.032709
2019-01-06	0.085096	-0.990868	0.0	7	6	6	6	-0.781831	0.848644	0.085731	-0.854975
2019-01-07	-0.066436	-0.989041	0.0	1	7	7	7	0.000000	0.937752	0.102821	-0.708966
2019-01-08	-0.104200	-0.987215	0.0	2	8	8	8	0.781831	0.988468	0.119881	-0.461984
2019-01-09	-0.017878	-0.985388	0.0	3	9	9	9	0.974928	0.998717	0.136906	-0.300453
2019-01-10	-0.270488	-0.983562	0.0	4	10	10	10	0.433884	0.968077	0.153891	-0.735951
2019-01-11	0.105717	-0.981735	0.0	5	11	11	11	-0.433884	0.897805	0.170830	-0.664646
2019-01-12	-0.100932	-0.979909	0.0	6	12	12	12	-0.974928	0.790776	0.187719	-1.079653
2019-01-13	-0.031559	-0.978082	0.0	7	13	13	13	-0.781831	0.651372	0.204552	-0.984968
2019-01-14	-0.029745	-0.976256	0.5	1	14	14	14	0.000000	0.485302	0.221325	-0.270694
2019-01-15	0.086681	-0.974429	0.0	2	15	15	15	0.781831	0.299363	0.238033	-0.448446
2019-01-16	-0.140585	-0.972603	0.0	3	16	16	16	0.974928	0.101168	0.254671	-0.670042
2019-01-17	0.042266	-0.970776	0.0	4	17	17	17	0.433884	-0.101168	0.271234	-0.727395
2019-01-18	0.129276	-0.968950	0.0	5	18	18	18	-0.433884	-0.299363	0.287717	-0.988036
2019-01-19	0.258936	-0.967123	0.0	6	19	19	19	-0.974928	-0.485302	0.304115	-1.093174
2019-01-20	-0.040826	-0.965297	0.0	7	20	20	20	-0.781831	-0.651372	0.320423	-1.376679

Trend + Tree-Based Ensemble Steps

Steps:

1. Fit trend to past (training data)
2. Predict trend into future (test data)
3. Detrend past (training data)
4. Fit tree-based model on detrended target calendar-based features
5. Predict tree-based model in future
6. Combine trend and tree-based prediction for final forecast

Trend Considerations:

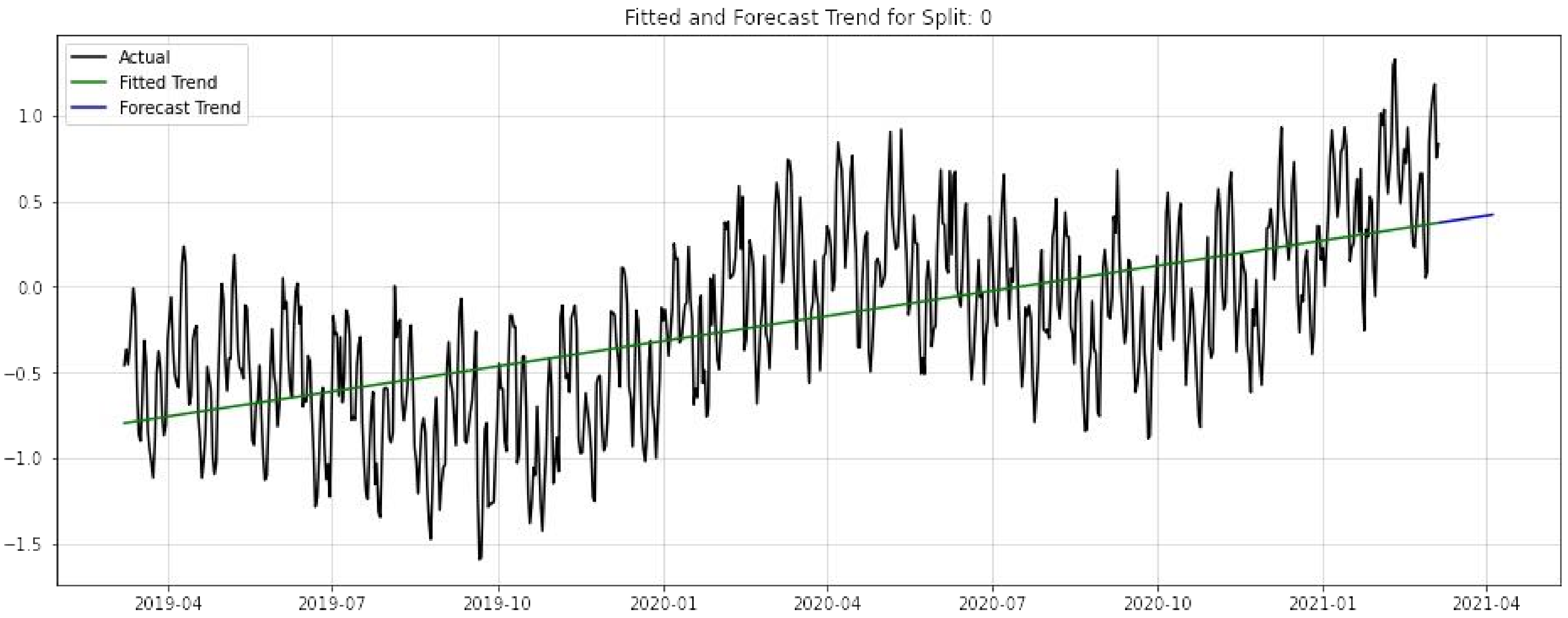
How to fit and forecast the trend?

- Linear Regression
- Polynomial Regression
- Spline/Loess/Kernel Regression
- Seasonal Decomposition
- Signal Filtering
- Do Nothing

Trend fit and forecast could be different methods

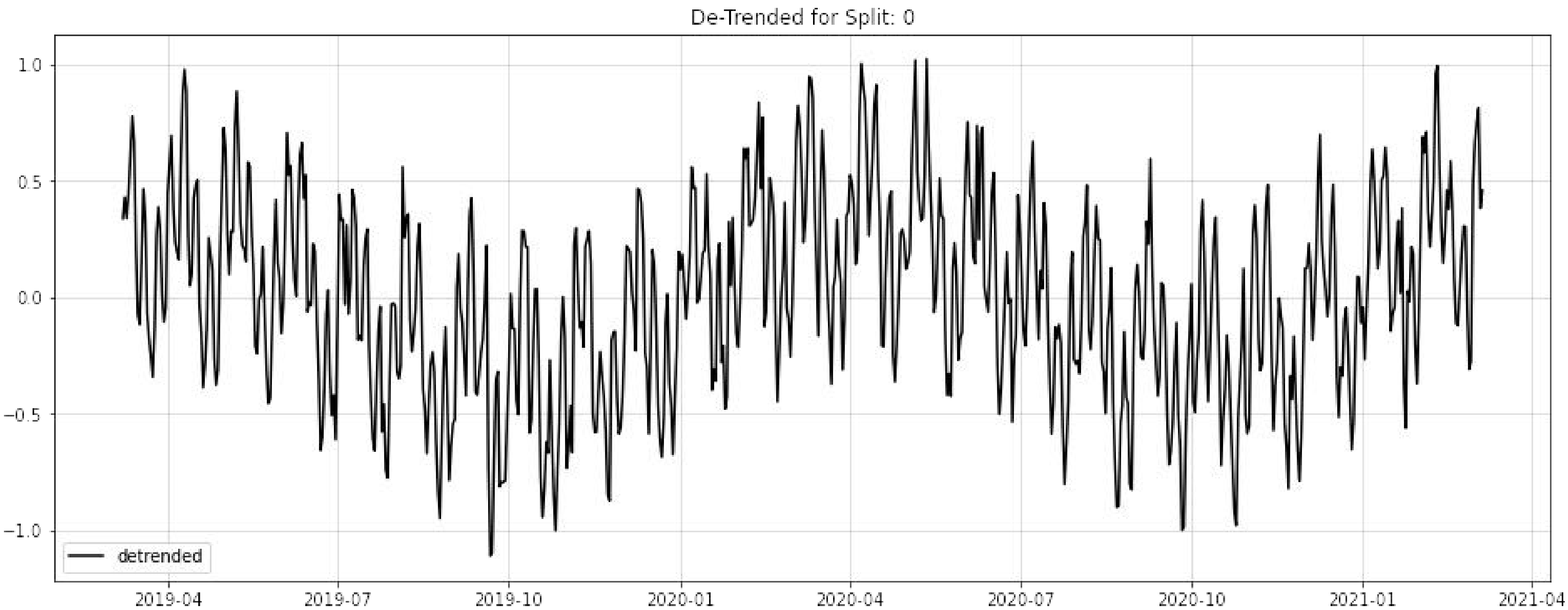
Fit and Forecast the Trend

LinearRegressor() from sklearn.linear_model



De-Trend the Target Variable

Detrend = Actual – Trend Fit

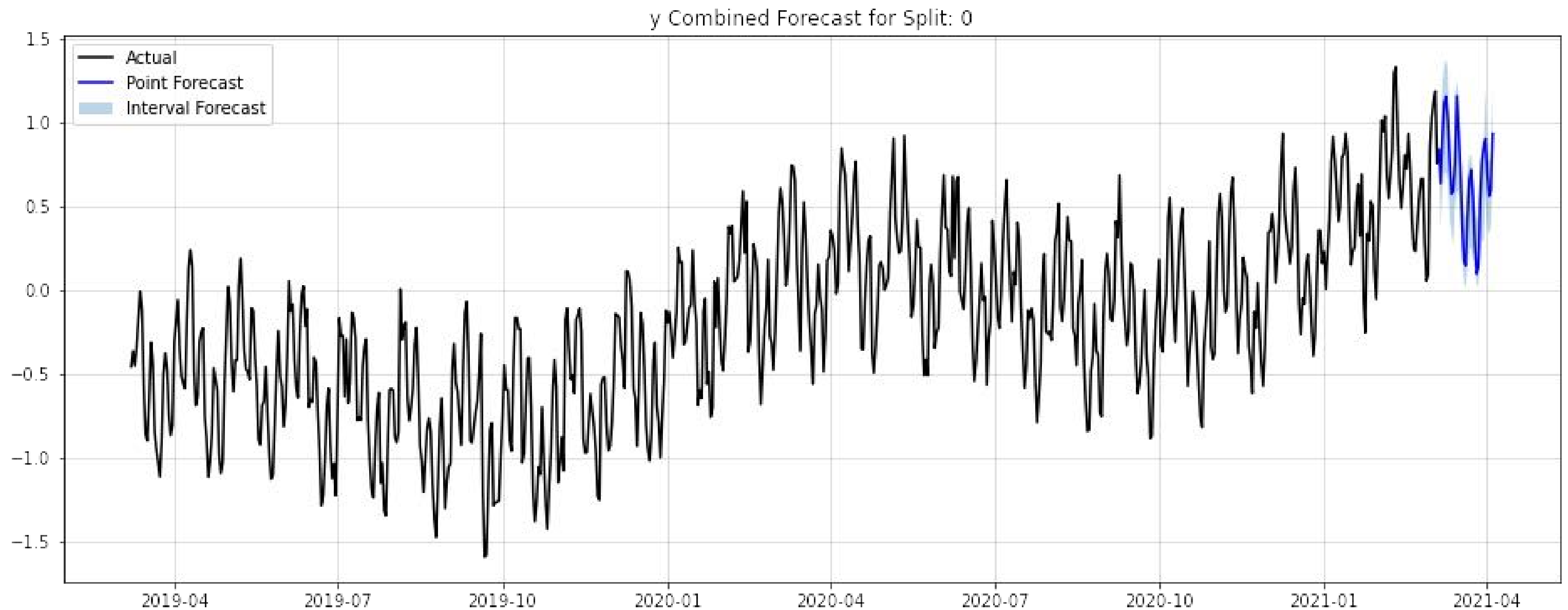


Train Tree-Based Model + Make Forecast

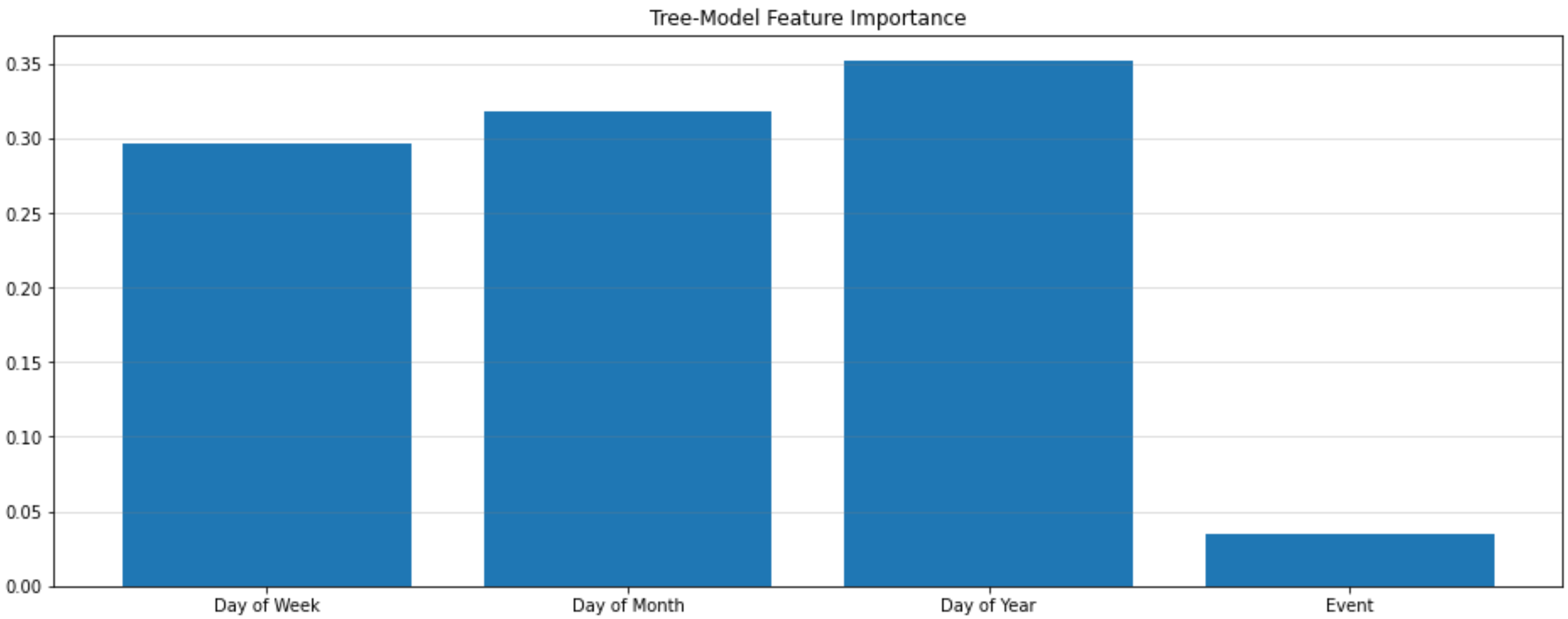
Final Forecast =

Trend Forecast (LinearRegressor) +

Tree Prediction (GradientBoostingRegressor)

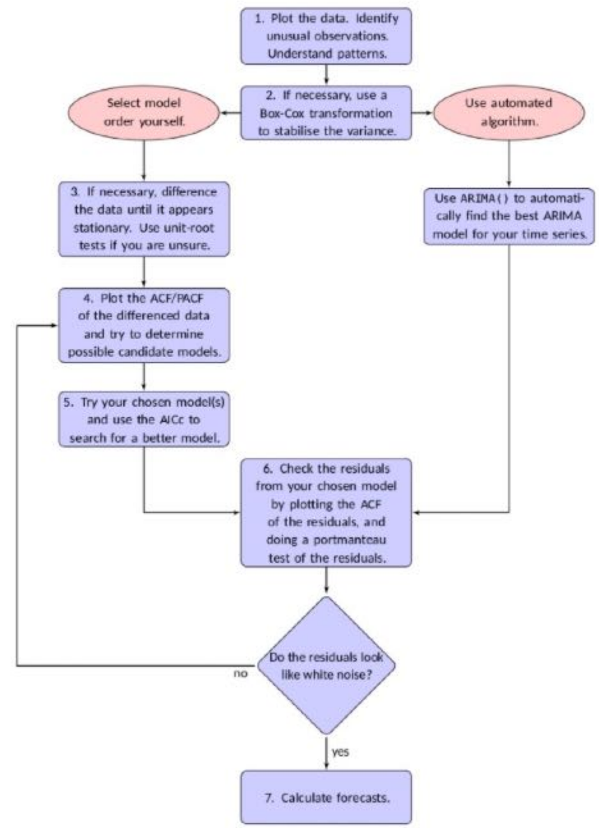


Feature Importance for Tree-Model

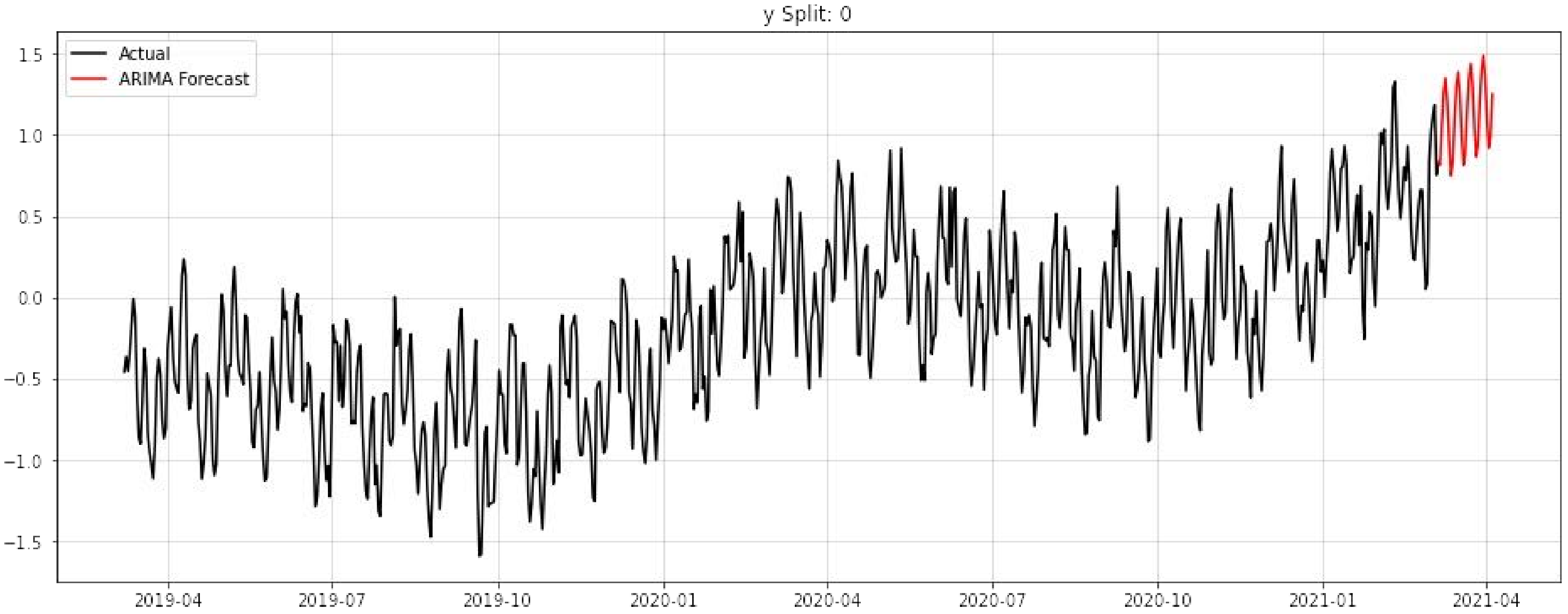


ARIMA

- ARIMA = Auto-Regressive Integrated Moving Average
- Use auto_arima() from pmdarima
  - See: https://alkaline-ml.com/pmdarima/tips_and_tricks.html
  - Also see Rob Hyndman's online textbook (FPP):
    - https://otexts.com/fpp2/arima-r.html (Version 2 - forecast::auto.arima)
    - https://otexts.com/fpp3/arima-r.html (Version 3 - fable::ARIMA)
- Searches for "best" ARIMA parameters (p, d, q)
- Note: set the "m" parameter (number of observations per seasonal cycle)
  - 7 for daily (used this for simulated data)
  - 12 for monthly
  - 52 for weekly



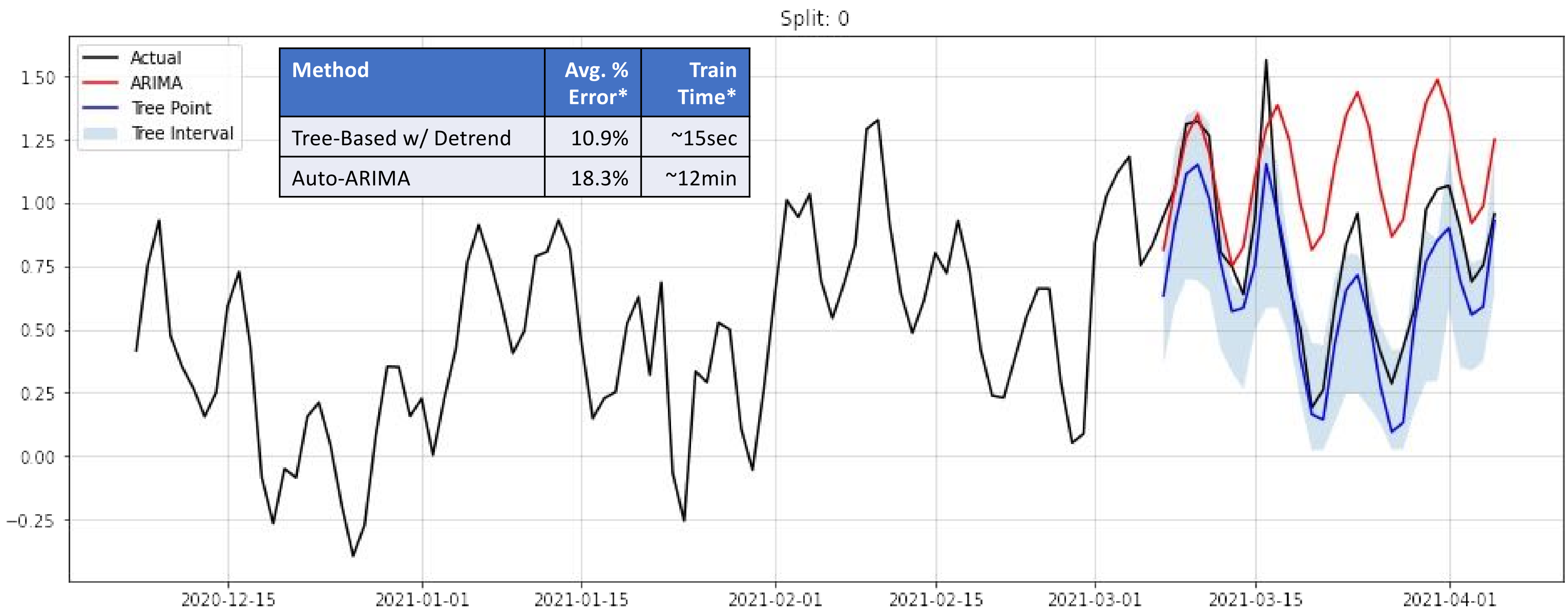
ARIMA Forecast



Trend + Tree-Based vs. (Auto) ARIMA

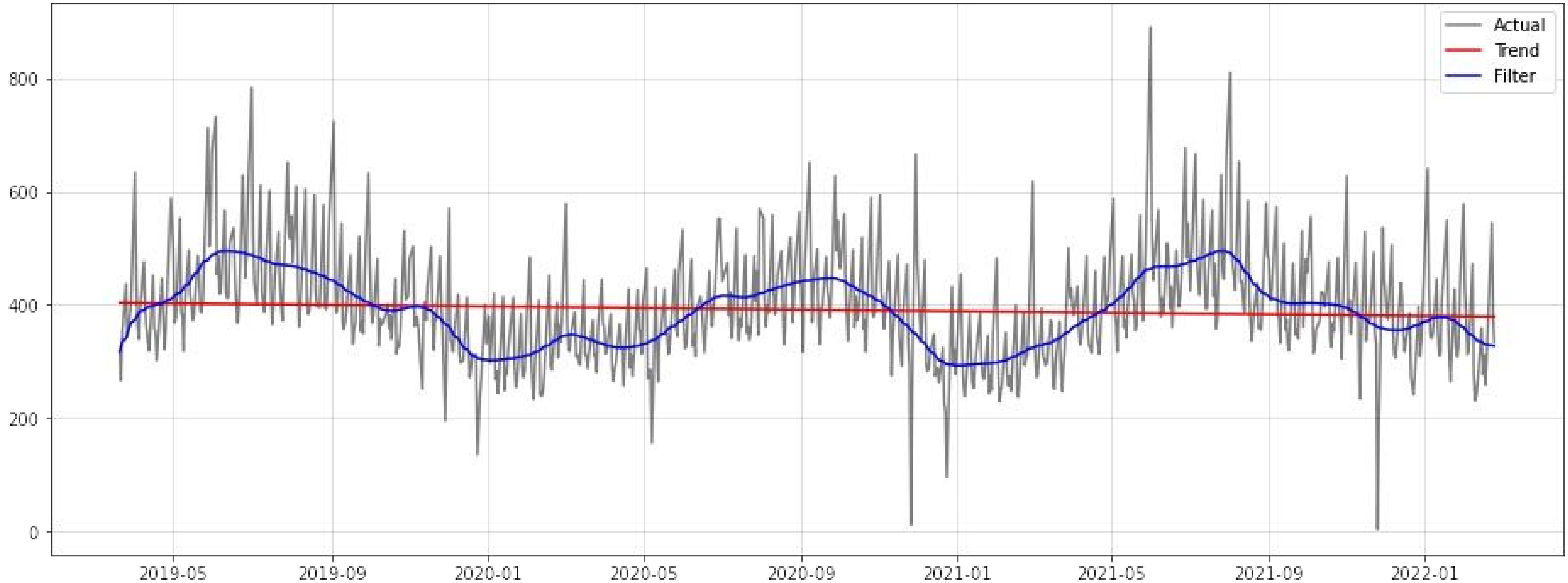
*Across 10 splits with:
test size = 30 and train size = 365*2
Using Lev's fin_err() function
[statistics/nb_Time_Series_Forecast_Error.ipynb at master · lselector/statistics · GitHub]

Method	Avg. % Error*	Train Time*
Tree-Based w/ Detrend	10.9%	~15sec
Auto-ARIMA	18.3%	~12min



Example Trend with Real-World Data Set

- Below is an example of real time series data set (blue line)
- Trend fit with linear regression (red line) vs. signal filter (blue line)



Conclusion

- Topic: use tree-base ensemble algo for time series forecasting (model trend + use calendar features)
- Hyper-parameter tuning of tree-based ensemble algo
  - Default params used in the demo
- Feature engineering options
  - Date/time components vs. trigonometric representation
  - Add other features that may be relevant to the time series being forecast
- Other considerations:
  - Regular vs. irregular time series
  - Ease of automation and/or implementation