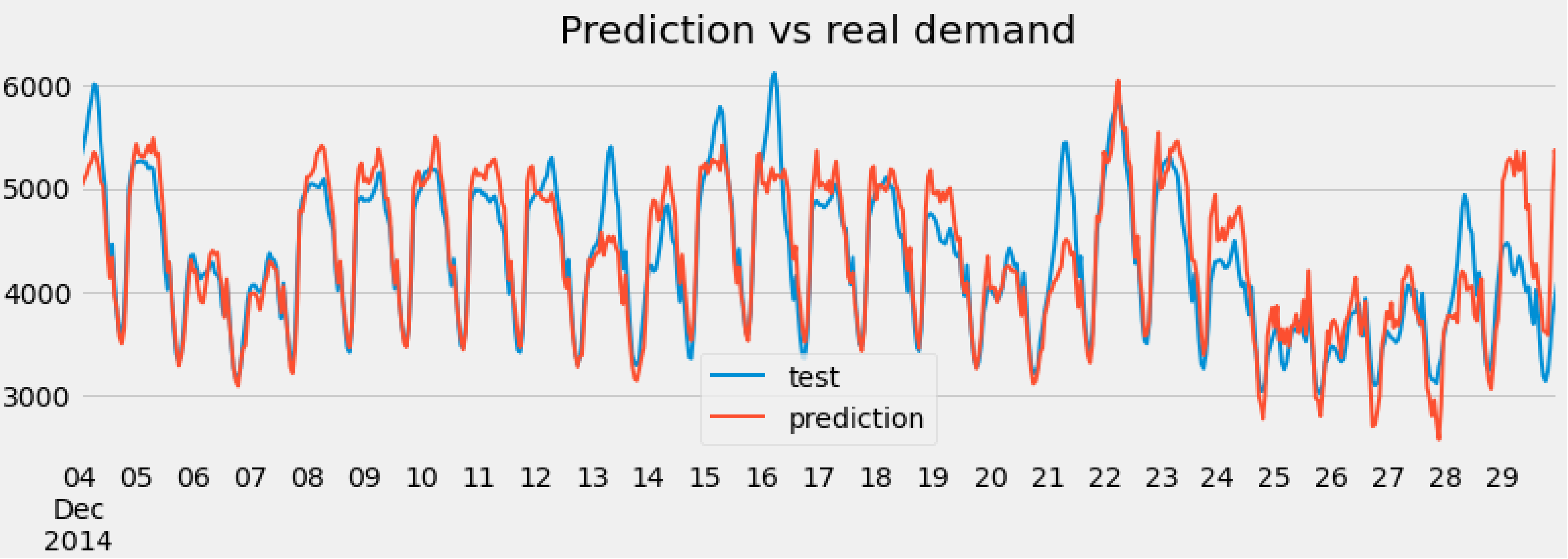


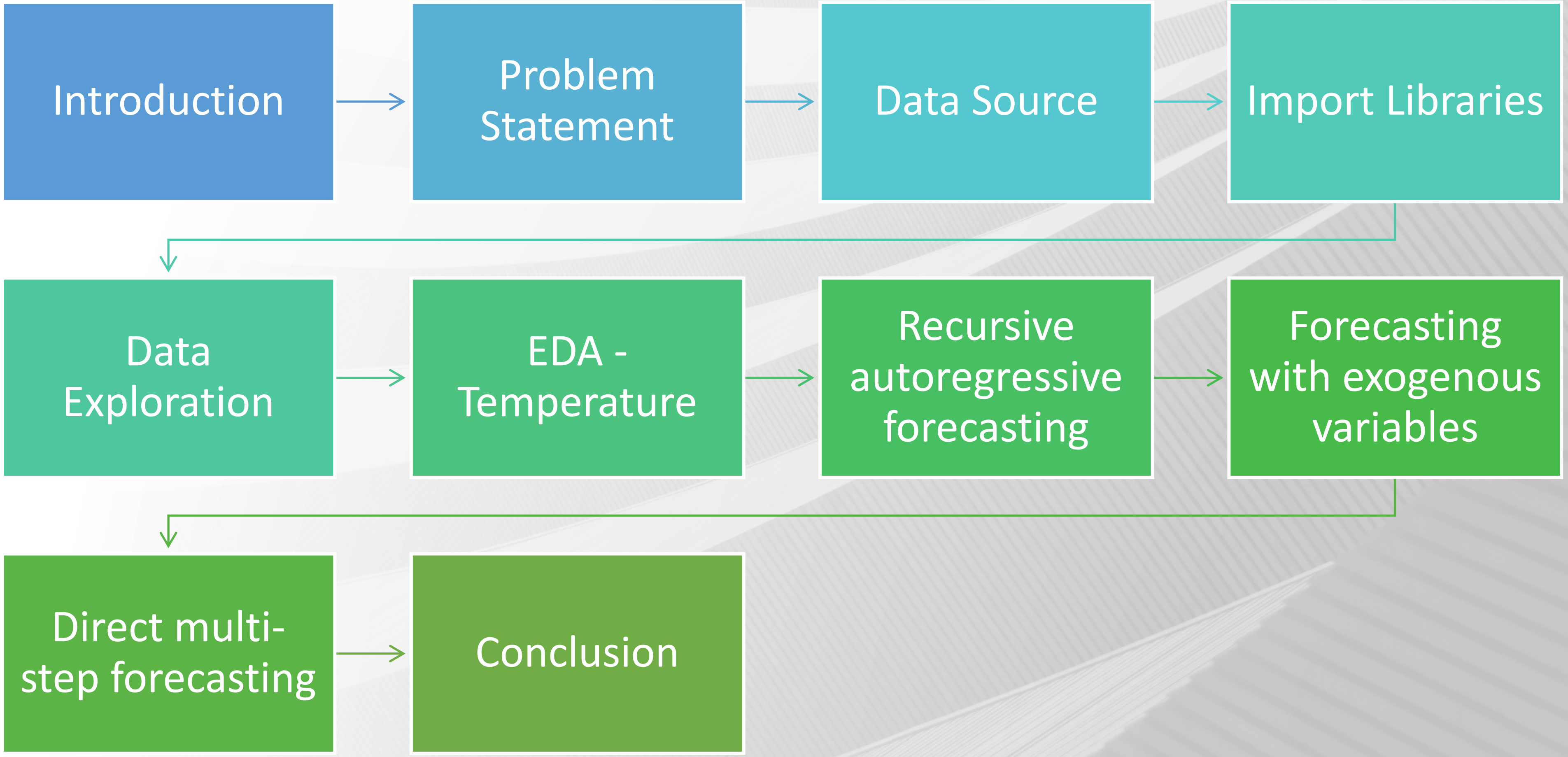
Forecasting electricity demand with Python

Olumide Aluko

MSDS 692 Data Science Practicum 1

June 23, 2022

Table of Content



Introduction

Problem Statement

Data Source

Import Libraries

Data Exploration

EDA - Temperature

Recursive autoregressive forecasting

Forecasting with exogenous variables

Direct multi-step forecasting

Conclusion

Project plan/overview

- A time series is a sequence of chronologically ordered data spaced at equal or unequal intervals.
- The forecasting method predicts a time series' future value by modeling it based on its past behavior (autoregressive) or by using other external variables.

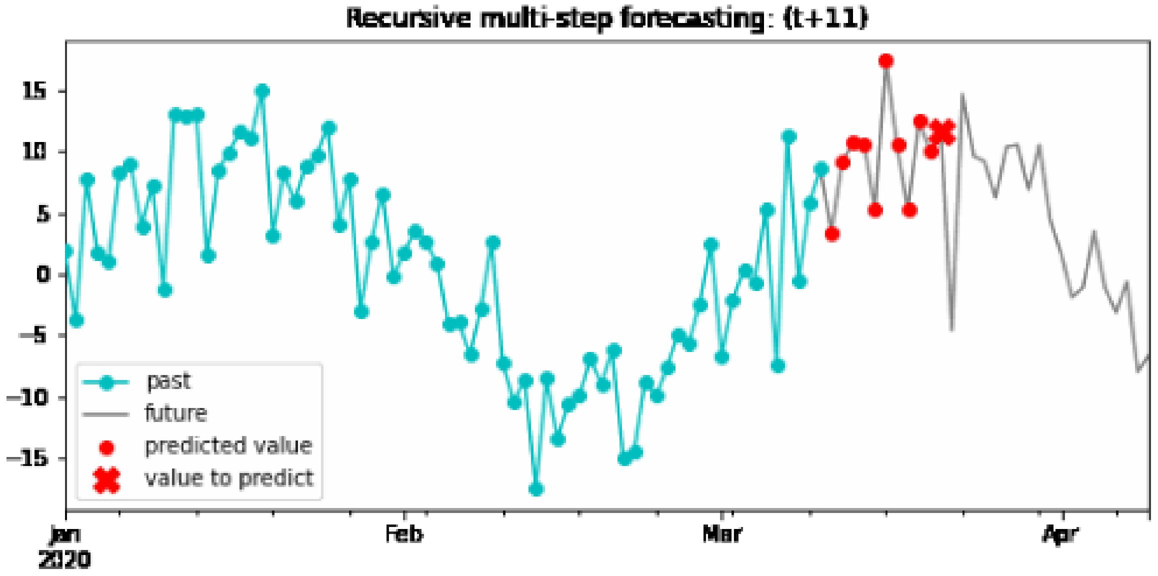


Fig 1

Problem Statement

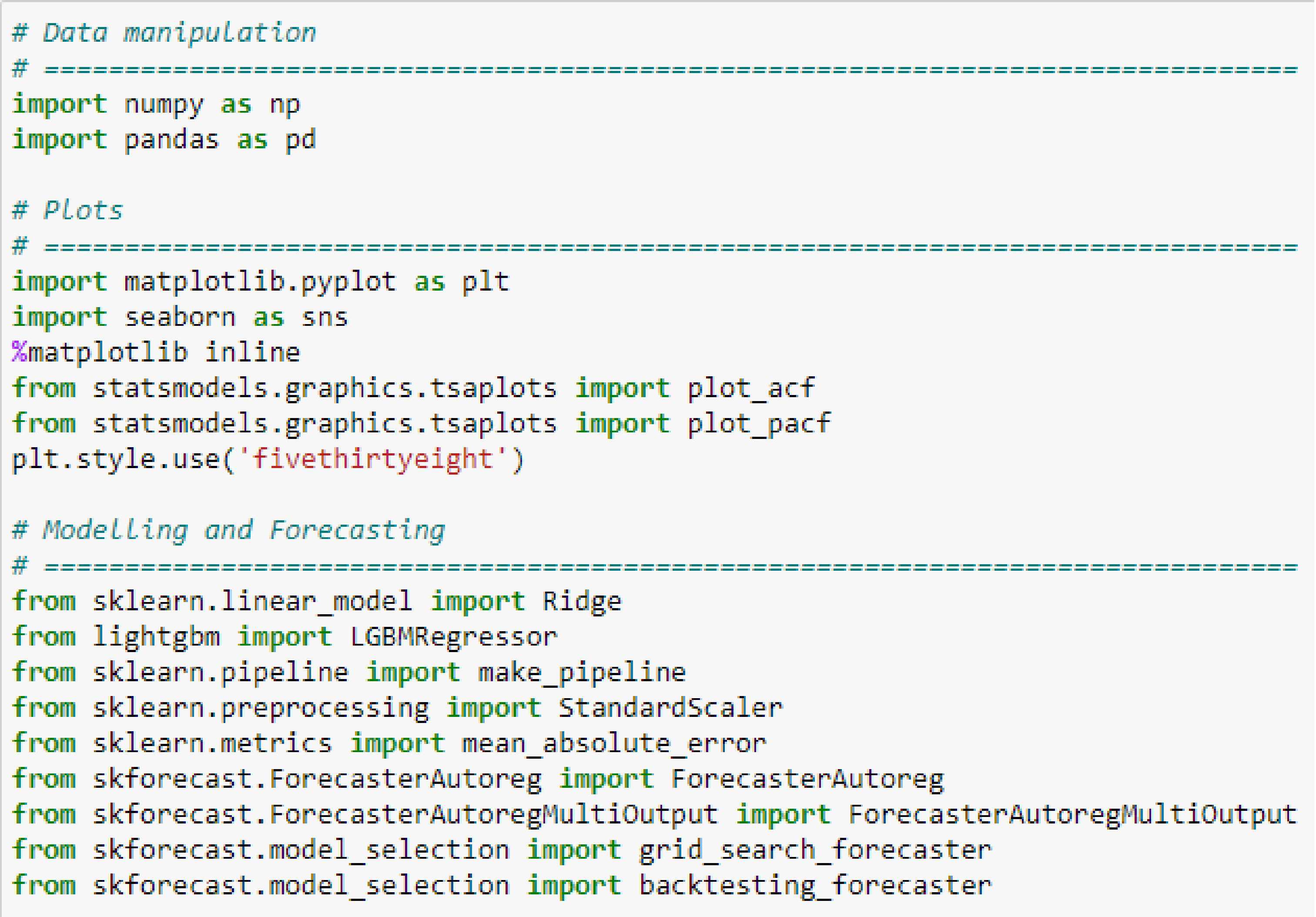
- A time series with electricity demand (Mega Watts) for the state of Victoria (Australia) from 2011-12-31 to 2014-12-31 is available.
- Demand for electricity in Australia has been in the spotlight for the general population due to the recently increasing price. Still, forecasts of the electricity demand have been expected to decrease due to various factors.
- The project aims to generate a forecasting model capable of predicting the next day's energy demand at the hourly level by accurately predicting monthly electricity demand.
- The proposed project design will be achieved using a time series forecasting with scikit-learn regressors

Fig 1

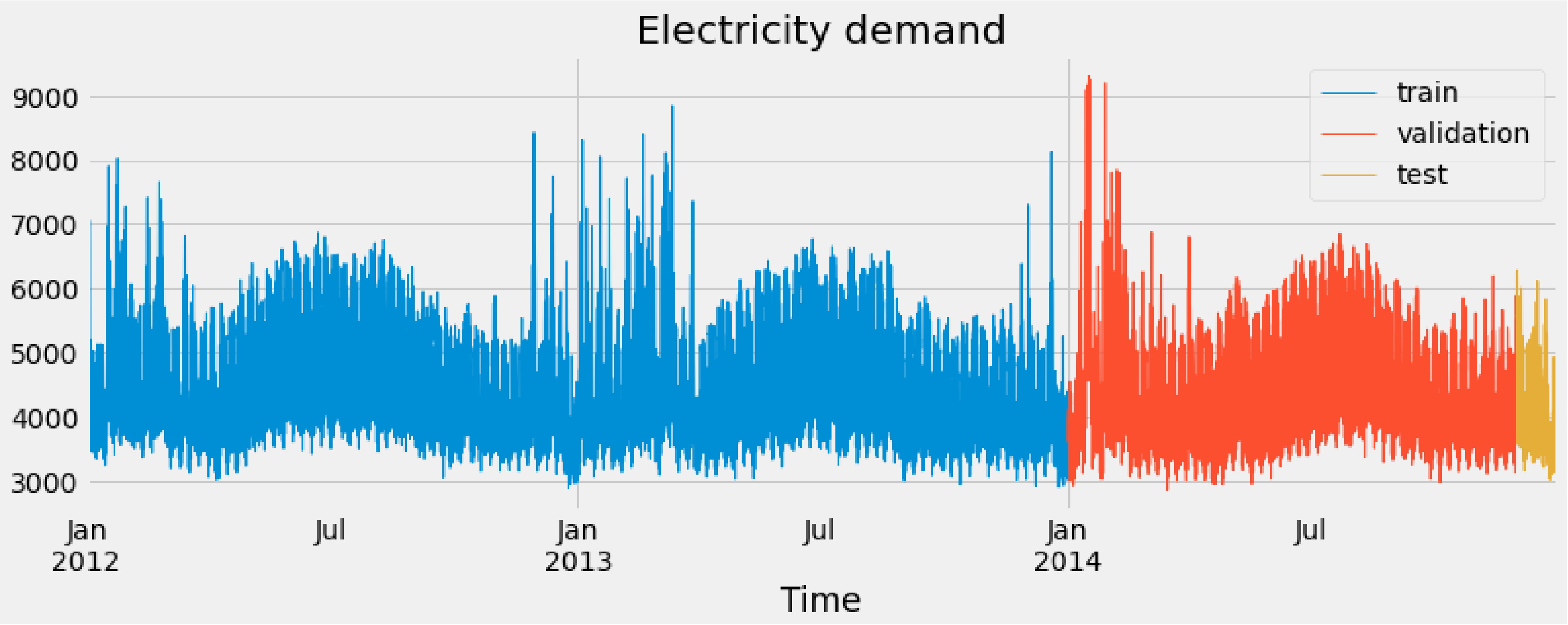
Dataset Source

- Dataset Source: https://raw.githubusercontent.com/JoaquinAmatRodrigo/skforecast/master/'+'data/vic_elec.csv’
- The dataset was downloaded from GitHub for this project. The dataset contains 5 columns and 52,608 complete records. The information in each column is:
- Time: date and time of the record.
- Date: date of the record.
- Demand: electricity demand (MW).
- Temperature: temperature in Melbourne, capital of the state of Victoria.
- Holiday: indicator if the day is a public holiday.

Libraries

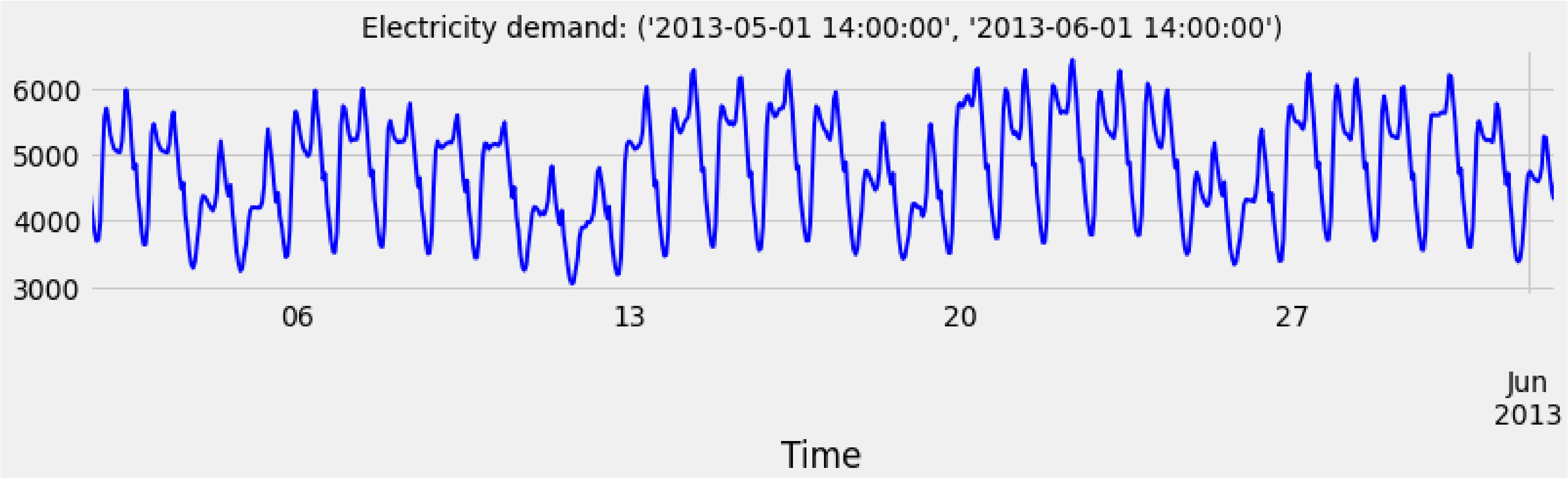


```
# Data manipulation
# ===============================================================================
import numpy as np
import pandas as pd

# Plots
# ===============================================================================
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
plt.style.use('fivethirtyeight')

# Modelling and Forecasting
# ===============================================================================
from sklearn.linear_model import Ridge
from lightgbm import LGBMRegressor
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error
from skforecast.ForecasterAutoreg import ForecasterAutoreg
from skforecast.ForecasterAutoregMultiOutput import ForecasterAutoregMultiOutput
from skforecast.model_selection import grid_search_forecaster
from skforecast.model_selection import backtesting_forecaster
```



Predicting Monthly Electricity Demand

- The graph shows that electricity demand has annual seasonality
- There is an increase centered on July and very accentuated demand peaks between January and March.

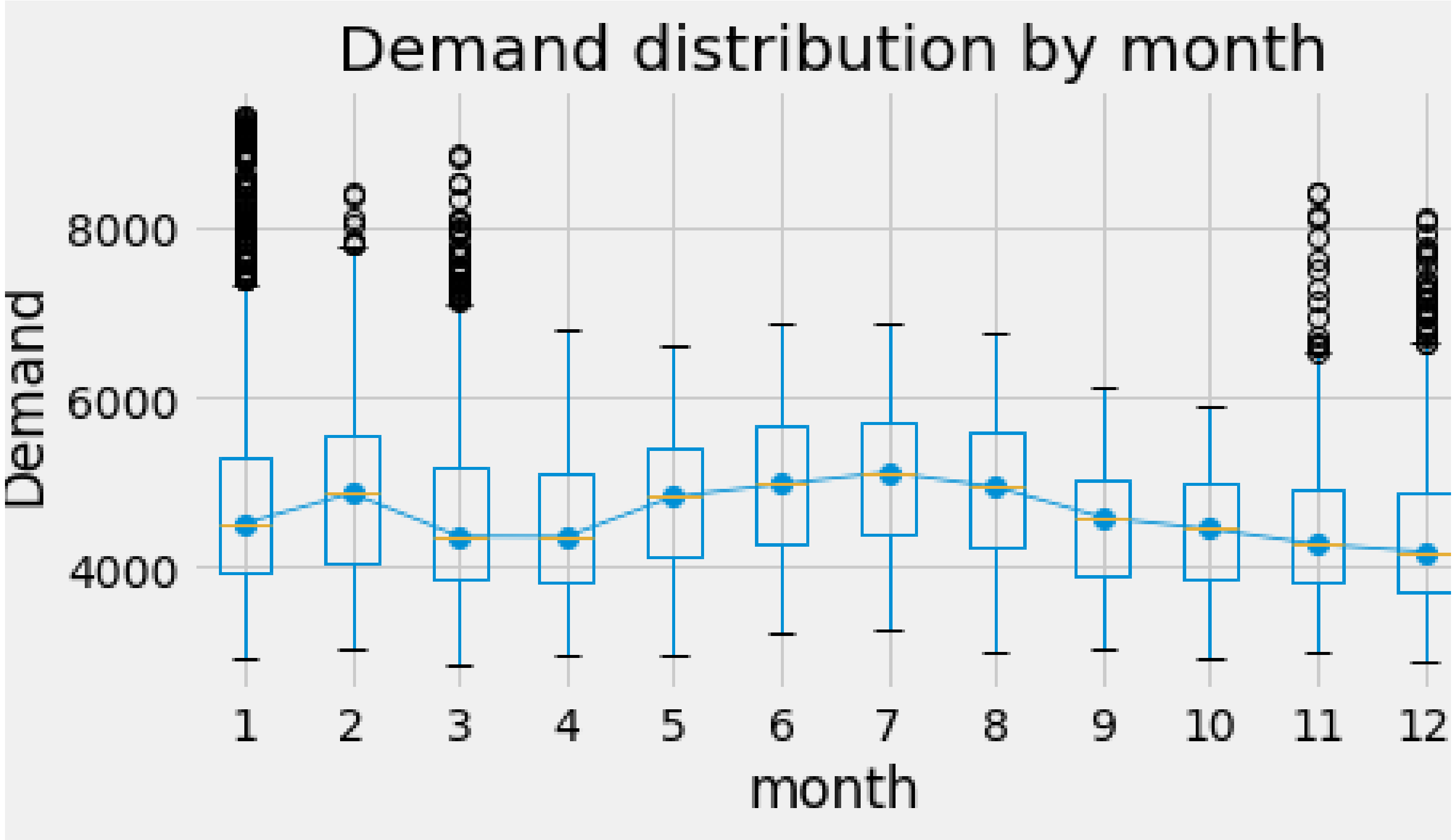


Section of the Electricity Demand Time Series

- When zooming in on the time series, a clear weekly seasonality is evident, with higher consumption during the work week (Monday to Friday) and lower consumption on weekends.
- It is also observed that there is a clear correlation between the consumption of one day and that of the previous day.

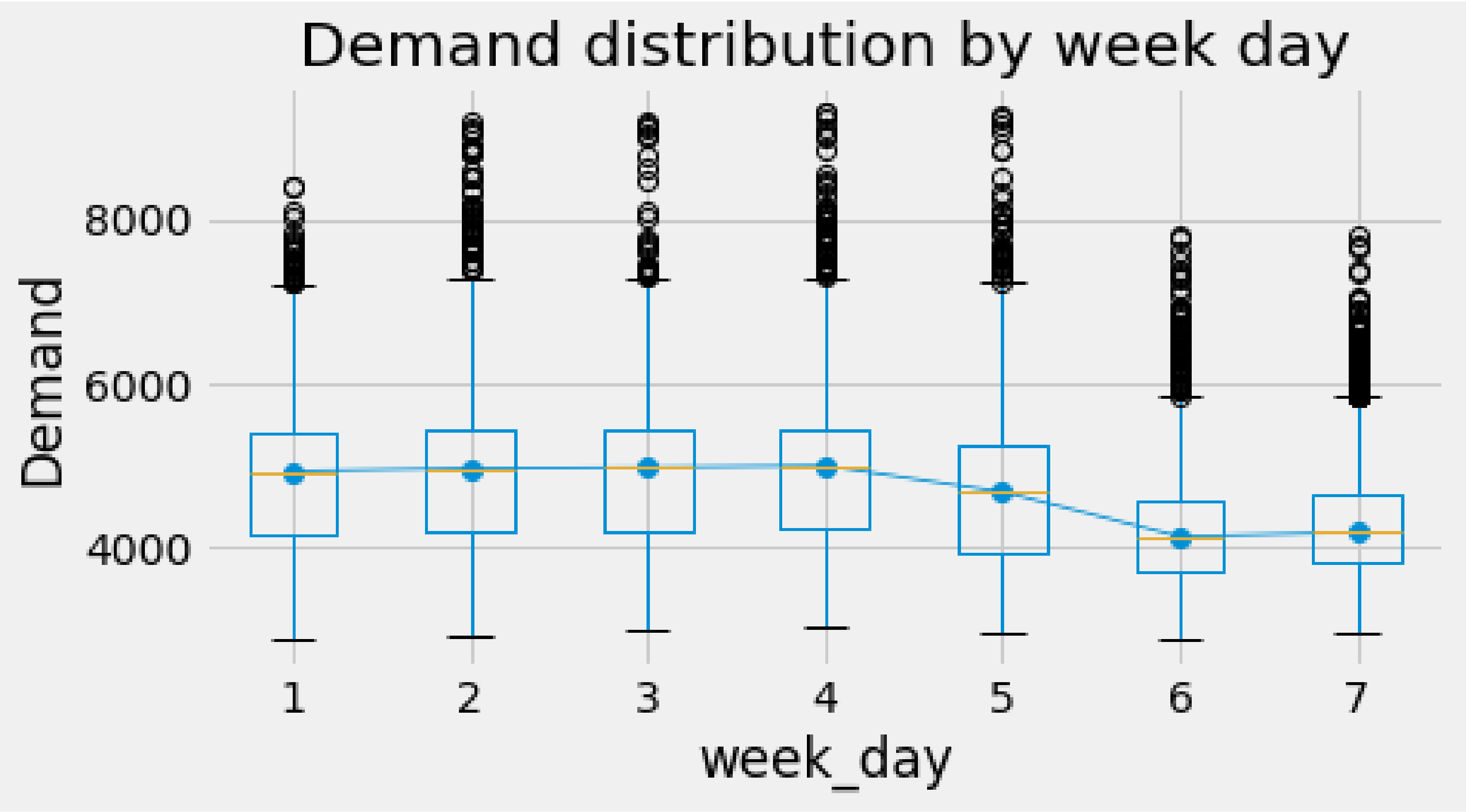
Electricity Demand Distribution by Month

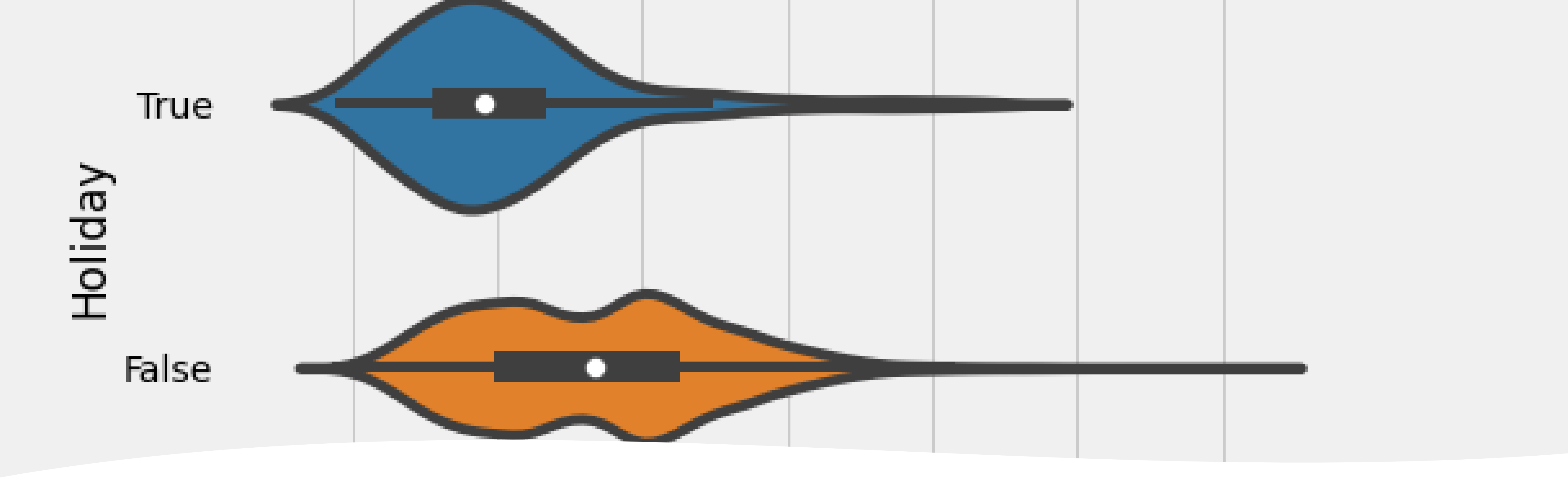
- It is observed that there is an annual seasonality, with higher (median) demand values in June, July, and August
- And with high demand peaks in November, December, January, February, and March



Demand Distribution by Weekdays

Weekly seasonality shows lower demand values during the weekend.



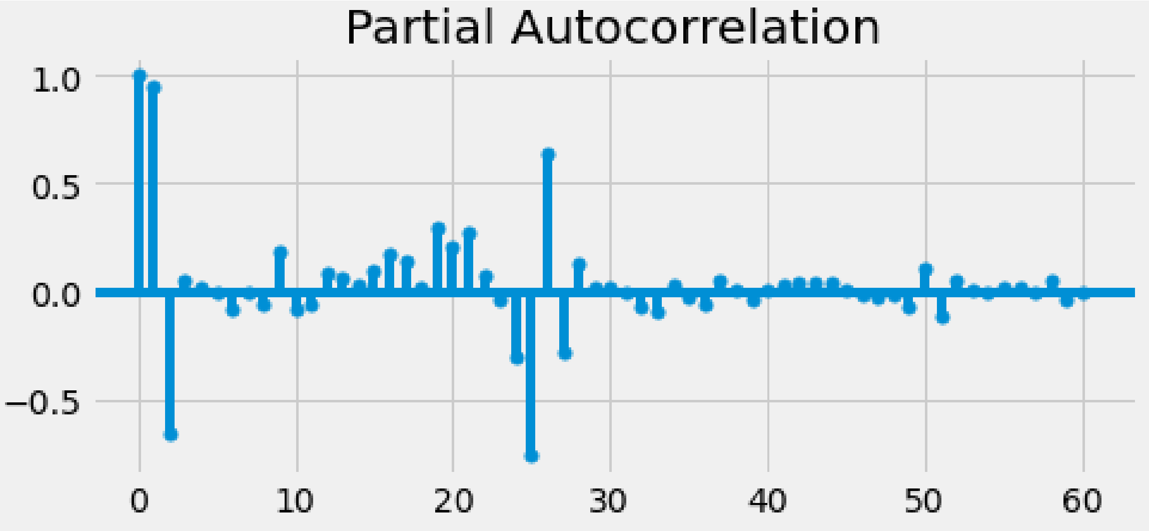


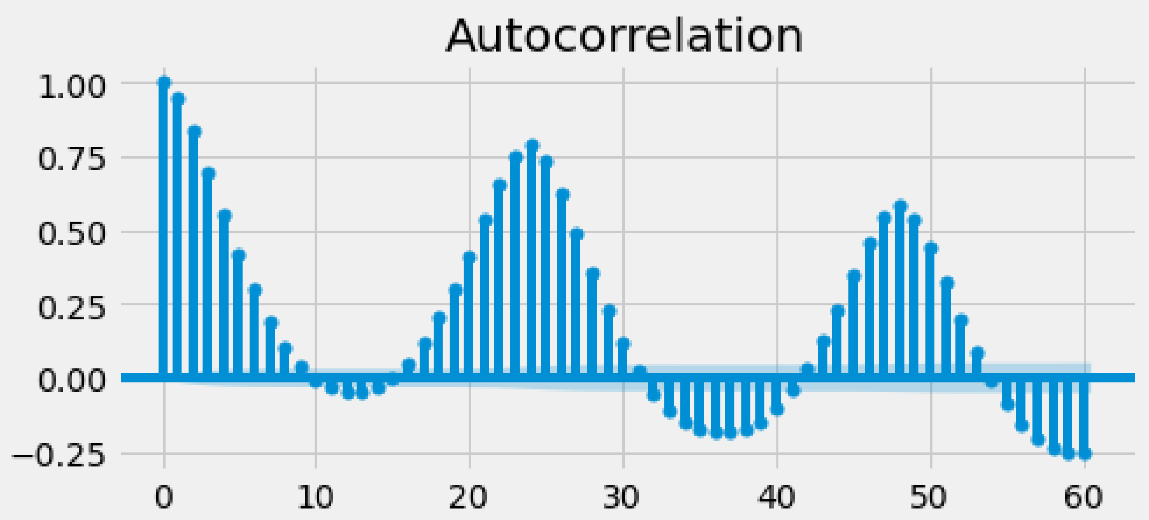
Holidays and Non-Holidays

- Holidays tend to have lower electricity consumption.

Partial Autocorrelation

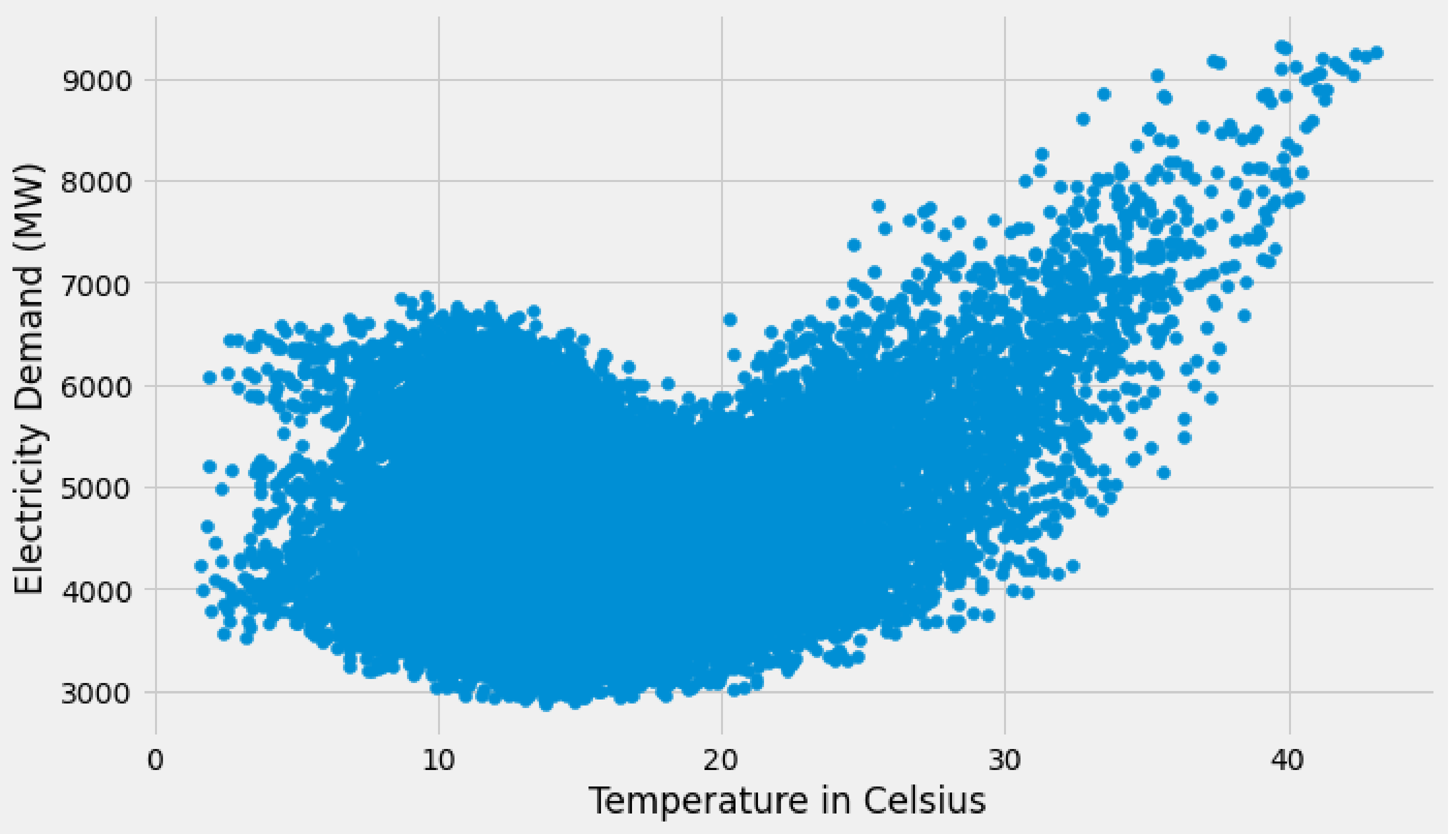
- The autocorrelation and partial autocorrelation plots show a clear association between one hour's demand and previous hours,
- As well as between one hour's demand and the same hour's demand on previous days
- This type of correlation is an indication that autoregressive models can work well for the forecast.





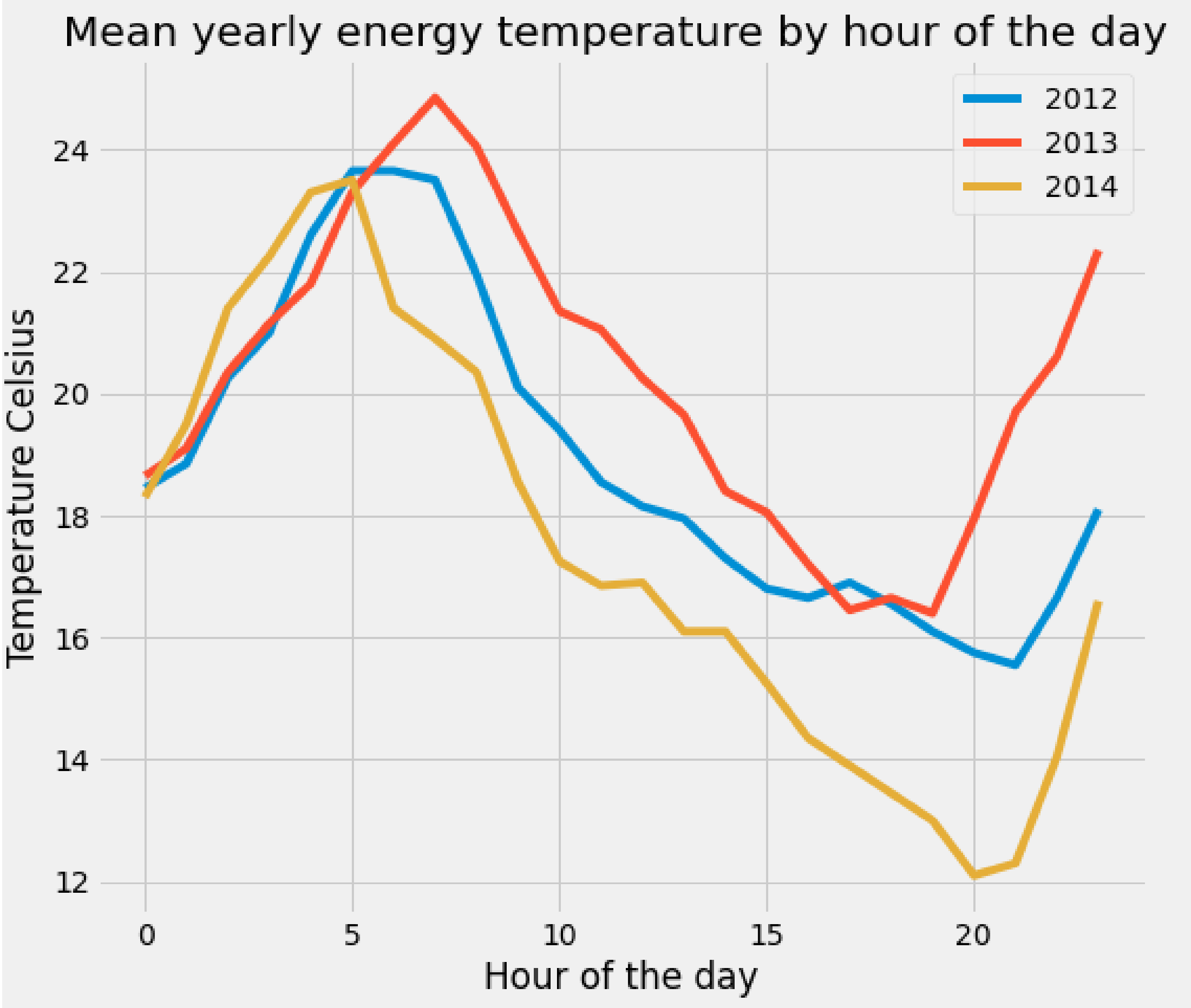
Correlation between Electricity Demand and the Temperature

- This scatterplot helps us to visualize the relationship between the variables.
- High demand occurs when temperatures are high because of air-conditioning. But there is also a heating effect, where demand increases for very low temperatures.



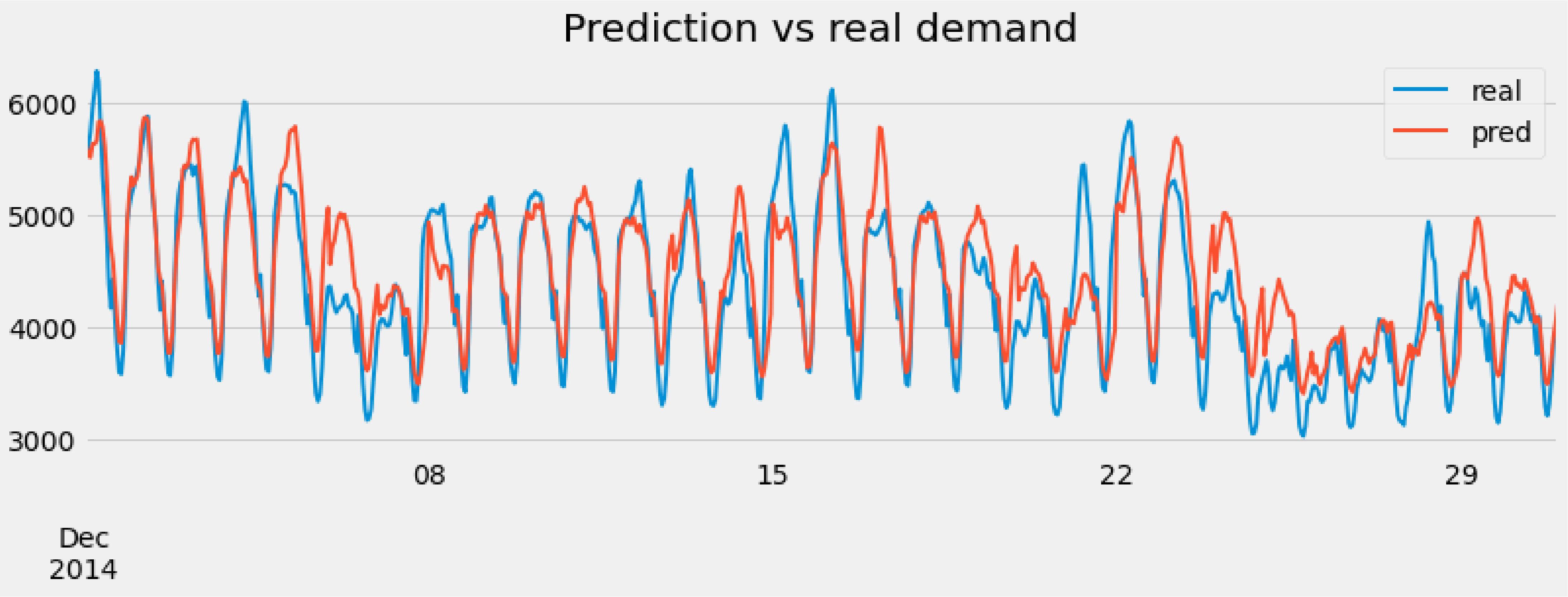
Mean Yearly Energy Demand by Hour of the Day

- From the visualization, we can see that the mean yearly energy temperature by hour of the day was highest in year 2013 at 0700 hour at temperature greater than 24 Celsius.



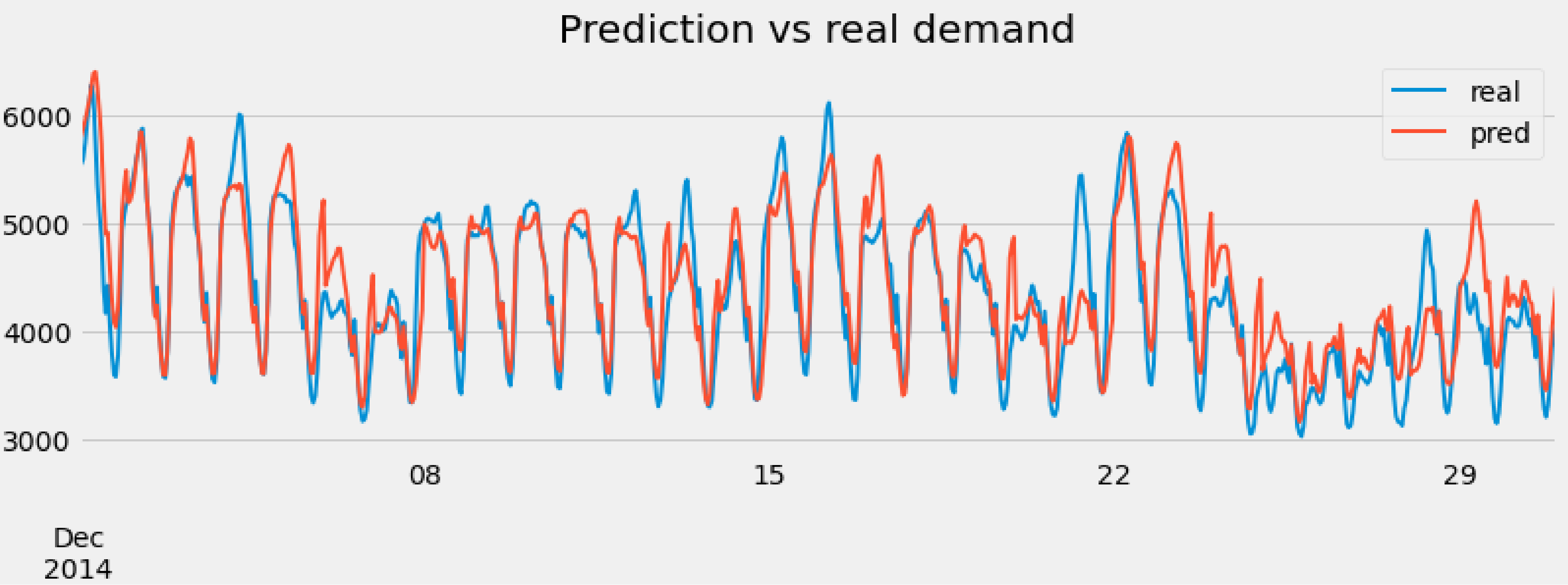
Recursive Autoregressive Forecasting

- A recursive autoregressive model is created and trained from a linear regression model with a Ridge penalty and a time window of 24 lags.
- The latter means that, for each prediction, the demand values of the previous 24 hours are used as predictors.
- Backtest explains how the model would have behaved if it had been trained with the data from 2012-01-01 00:00 to 2014-11-30 23:59 and then, at 23:59 each day, the following 24 hours were predicted is evaluated
- Backtest error result was equal to 289.5



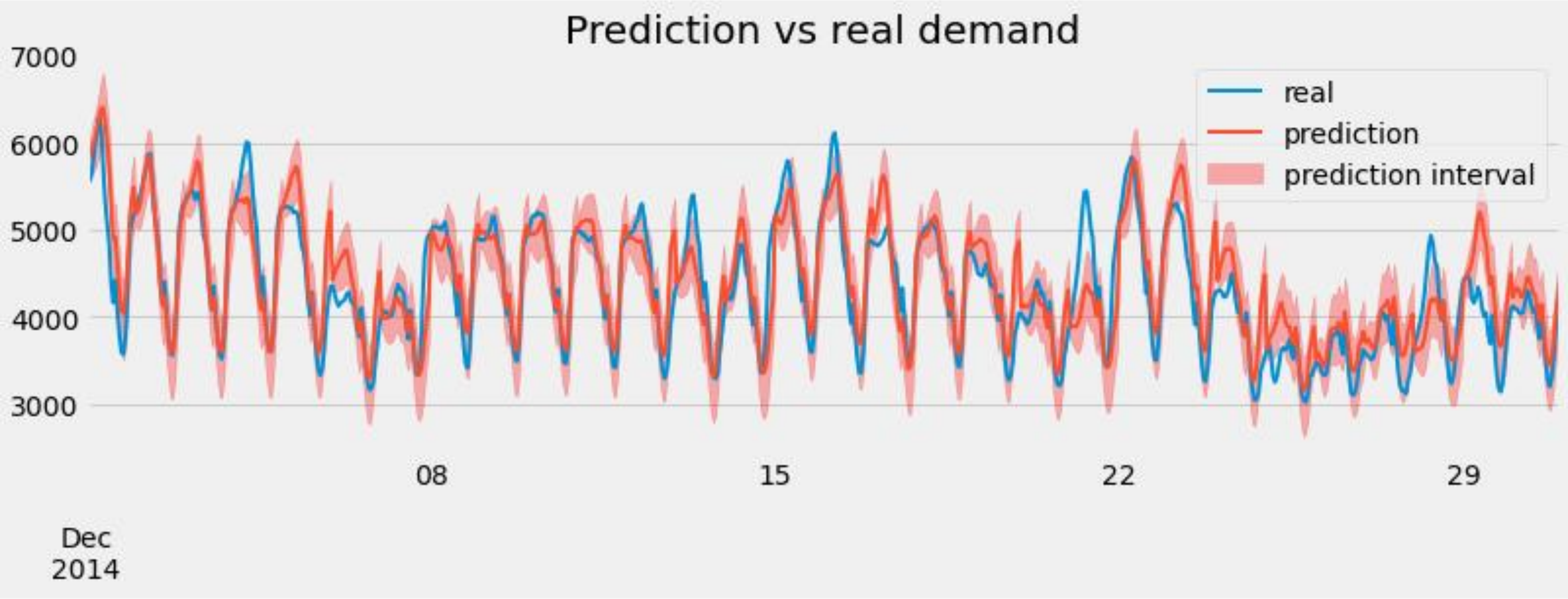
Hyperparameter Tuning

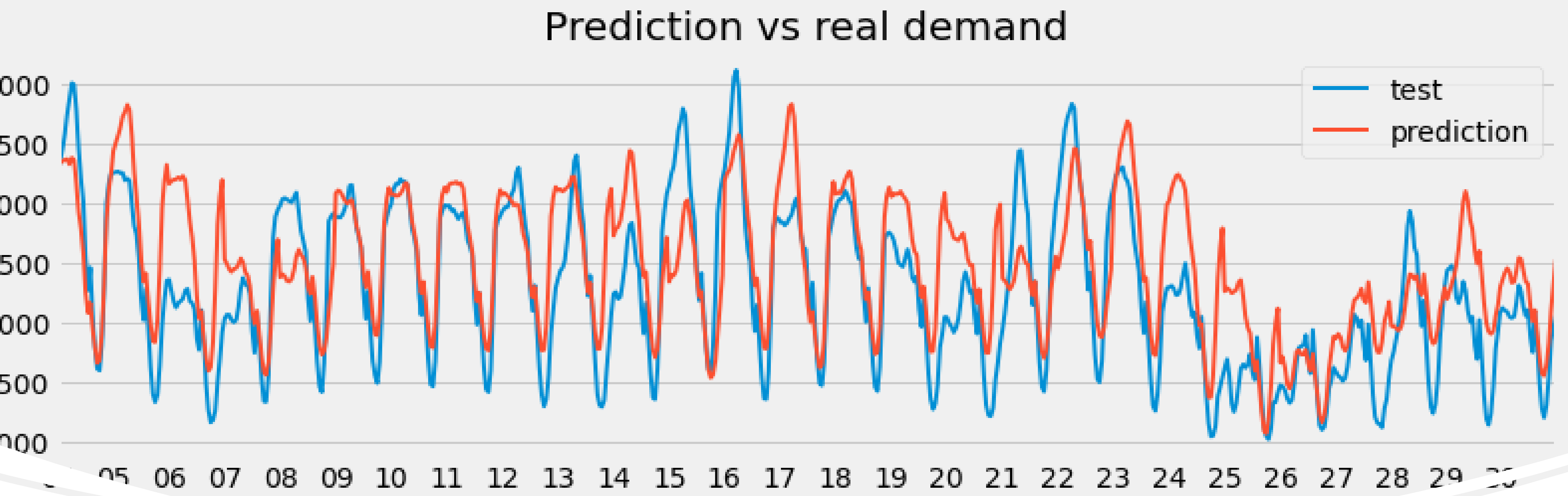
- To recognize the best combination of lags and hyperparameters, a Grid Search with validation by Backtesting is employed
- In the trained ForecasterAutoreg object, the first 24 lags and a Ridge model with the default hyperparameters have been utilized
- The best results are obtained by using the lags [1, 2, 3, 23, 24, 25, 47, 48, 49] and a Ridge configuration {'alpha': 215.44}
- Backtest with test data - once the best model has been identified and trained, its error in predicting the test data is calculated.
- After optimizing lags and hyperparameters, I observed that the prediction error was reduced from 289.5 to 251.9.



Prediction Intervals

- A prediction interval defines the interval within which the true value of "y" can be expected to be found with a given probability
- The predicted interval has a lower coverage than expected (80%).
- It may be due to the marked high error made by the model for days 21, 24, and 25.
- These days are within the Christmas holiday period, usually characterized by a different consumption behavior than the rest of the month.



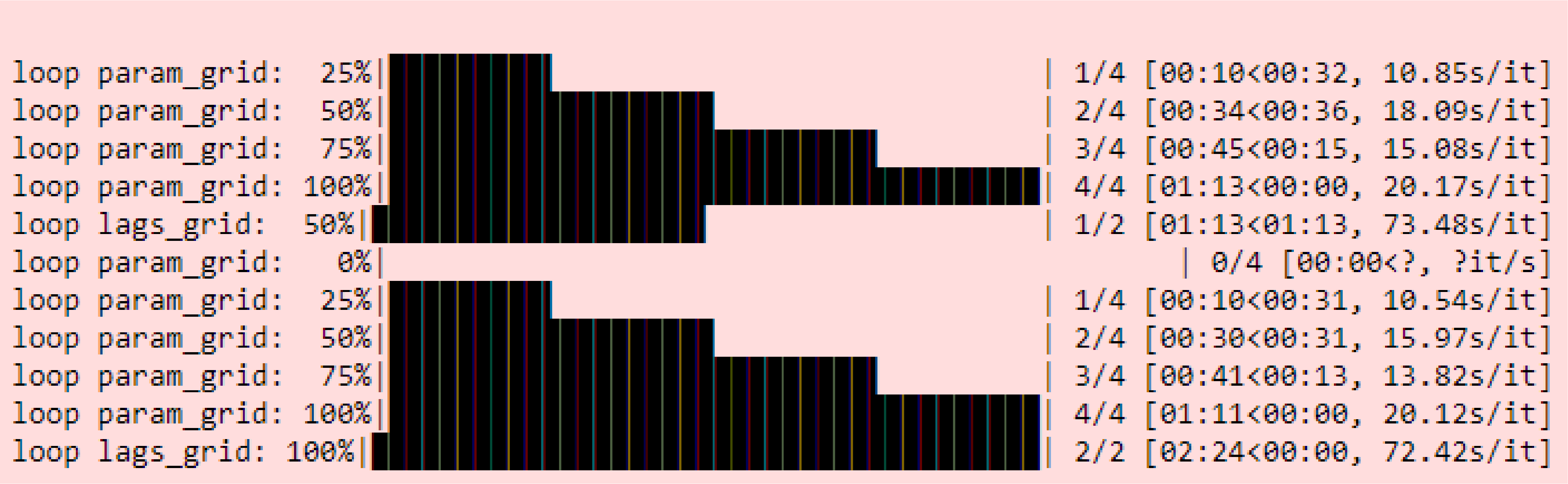


Forecasting with Exogenous Variables

- We use variables whose future value is known and can serve as additional predictors in the model, for example holidays, the month of the year, day of the week & time of day.
- By using as an exogenous variable whether the day is a holiday or not, the prediction error is slightly reduced from 394.5 to 375.1.
- The new predictors further reduce the model error with a Backtest error equal to 311.3

Direct Multi-Step Forecasting

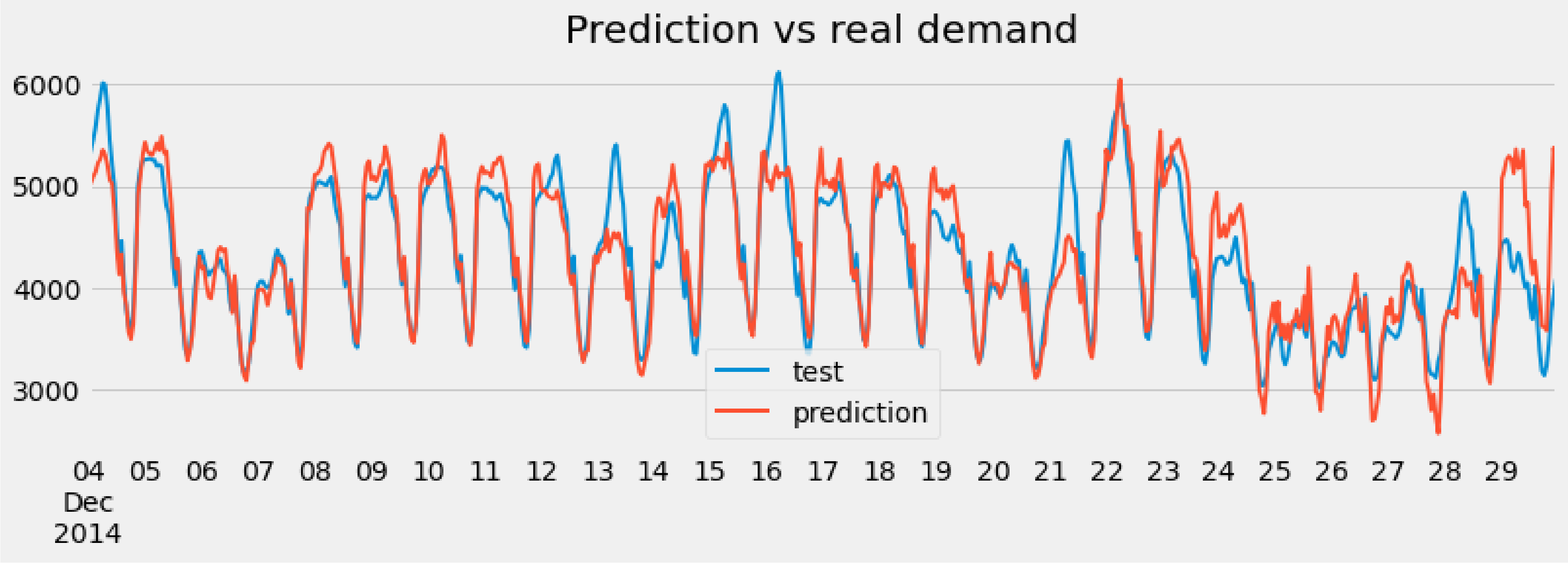
- The ForecasterAutoreg and ForecasterAutoregCustom models follow a recursive prediction strategy in which each new prediction builds on the previous one
- Forecaster` refitted using the best-found lags and parameters, and the whole data set: Lags: [1 2 3 23 24 25 47 48 49]
- Backtesting metric: 237.27478811428122



```
loop param_grid:  25%|██████████                              | 1/4 [00:10<00:32, 10.85s/it]
loop param_grid:  50%|████████████████████                    | 2/4 [00:34<00:36, 18.09s/it]
loop param_grid:  75%|██████████████████████████████          | 3/4 [00:45<00:15, 15.08s/it]
loop param_grid: 100%|████████████████████████████████████████| 4/4 [01:13<00:00, 20.17s/it]
loop lags_grid:  50%|████████████████████                     | 1/2 [01:13<01:13, 73.48s/it]
loop param_grid:   0%|                                                | 0/4 [00:00<?, ?it/s]
loop param_grid:  25%|██████████                              | 1/4 [00:10<00:31, 10.54s/it]
loop param_grid:  50%|████████████████████                    | 2/4 [00:30<00:31, 15.97s/it]
loop param_grid:  75%|██████████████████████████████          | 3/4 [00:41<00:13, 13.82s/it]
loop param_grid: 100%|████████████████████████████████████████| 4/4 [01:11<00:00, 20.12s/it]
loop lags_grid: 100%|█████████████████████████████████████████| 2/2 [02:24<00:00, 72.42s/it]
```

Anticipated Daily Forecast

- The backtesting process, including the new exogenous variables, is repeated
- In this case, using a gradient boosting LGBMRegressor model, the direct multi-step strategy generates better results than the recursive strategy with a linear model.
- Th Backtest error is equal to 228.2.

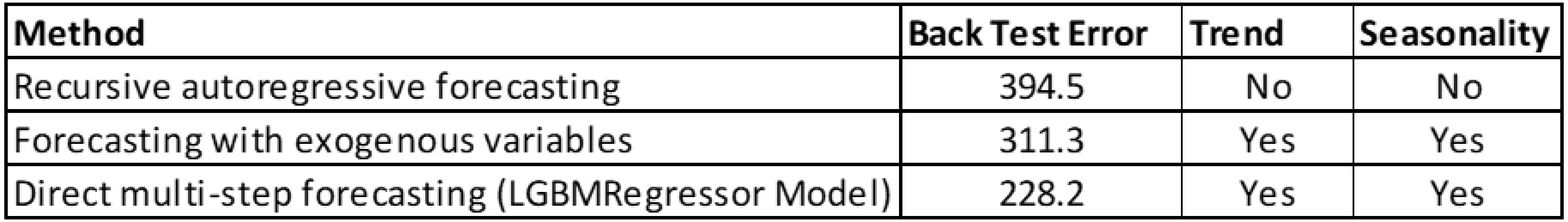


Conclusion

In this project, I used time-series data with electricity demand (MW) for the state of Victoria (Australia) from 2011-12-31 to 2014-12-31 to generate a forecasting model that predicts the next day's energy demand at the hourly level.

I started with exploratory data analysis in order to explore the characteristics of our dataset. In data cleaning process, null values are replaced with mean value and some attributes are extracted since they were redundant.

Hourly Electricity Demand are aggregated to daily level, so we can see the trend and seasonality functions more clearly. The dataset is split into 2 sets, train and test, in order to train the models and test our results.



Method	Back Test Error	Trend	Seasonality
Recursive autoregressive forecasting	394.5	No	No
Forecasting with exogenous variables	311.3	Yes	Yes
Direct multi-step forecasting (LGBMRegressor Model)	228.2	Yes	Yes

References

- https://machinelearningmastery.com/multi-step-time-series-forecasting-with-machine-learning-models-for-household-electricity-consumption/

- https://rstudio-pubs-static.s3.amazonaws.com/430567_e3e7a704a4dd45bbbb01abd7f03c7383.html

- https://jasondeegan.com/forecasting-norwegian-energy-demand-with-python/

- https://www.machinelearningplus.com/time-series/time-series-analysis-python/

- https://www.cienciadedatos.net/py29-forecasting-electricity-power-demand-python.html