AI 620 Emerging Topics in Artificial Intelligence

HOS09A DeepAR Models

03/28/2023 Developed by Yared Shewarade

09/23/2024 Updated by Anh Nguyen

11/19/2024 Reviewed by Jonathan Koerber

School of Technology and Computing (STC) @City University of Seattle (CityU)

Before You Start

- The directory path shown in screenshots may be different from yours.
- Some steps are not explained in the tutorial**.** If you are not sure what to do:
    1. Consult the resources listed below.
    2. If you cannot solve the problem after a few tries, ask the courses student worker for help.

Learning Outcomes

Students will be able to learn:

- Introduction to DeepAR Models
- Time series forecasting with DeepAR

Resources

- Tripuraneni, S., & Song, C. (2019). *Hands-on artificial intelligence on amazon web services: Decrease the time to market for AI and ML applications with the power of AWS* (1st ed.). Packt.

Introduction to DeepAR Models

DeepAR, which is offered by Amazon SageMaker, is a deep learning algorithm based on the recurrent neural networks designed specifically for time series forecasting. It is a supervised learning algorithm for forecasting scalar (one dimensional) time series that could estimate the future probability distribution instead of a specific number. Time series forecasting use cases are certainly the most common time series use cases, as they can be found in all types of industries and in various contexts. Whether it is forecasting future sales to optimize inventory, predicting energy consumption to adapt production levels, or estimating the number of airline passengers to ensure high-quality services, time is a key variable.

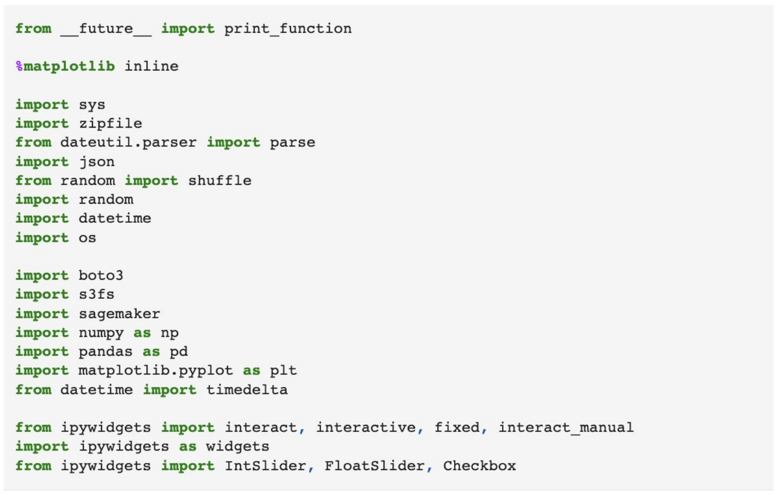
DeepAR is a forecasting method based on autoregressive neural networks, and it learns about a global model from historical data of all-time series in the data set. DeepAR employs Long Short-

Term Memory **(**LSTM**),** a type of Recurrent Neural Network **(**RNN**)**, to model time series. The main idea of RNNs is to capture sequential information.

Time series forecasting with DeepAR

Note: Open Notebook using SageMaker. Follow the steps in HOS07 to open a Notebook.

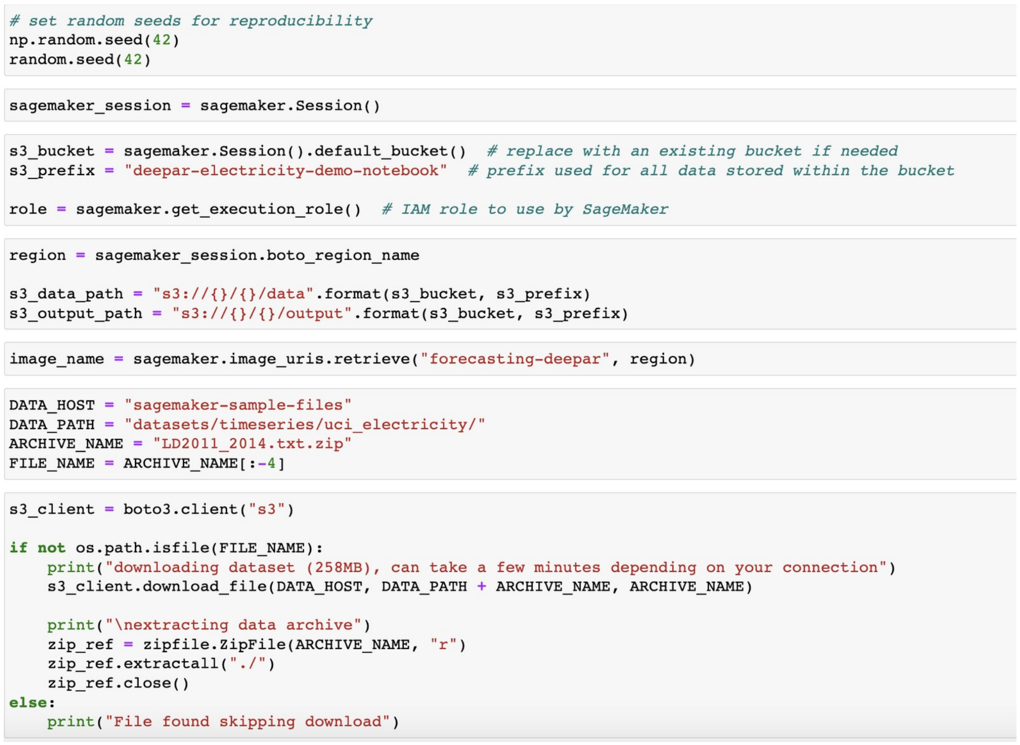
A. Install the required module and prepare the data.



```
from __future__ import print_function

%matplotlib inline

import sys
import zipfile
from dateutil.parser import parse
import json
from random import shuffle
import random
import datetime
import os

import boto3
import s3fs
import sagemaker
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import timedelta

from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets
from ipywidgets import IntSlider, FloatSlider, Checkbox
```

Let’s download the data and transfer to S3.

Note:- The dataset is provided with this file in GitHub or download from here, upload to your notebook.



```
# set random seeds for reproducibility
np.random.seed(42)
random.seed(42)
```

```
sagemaker_session = sagemaker.Session()
```

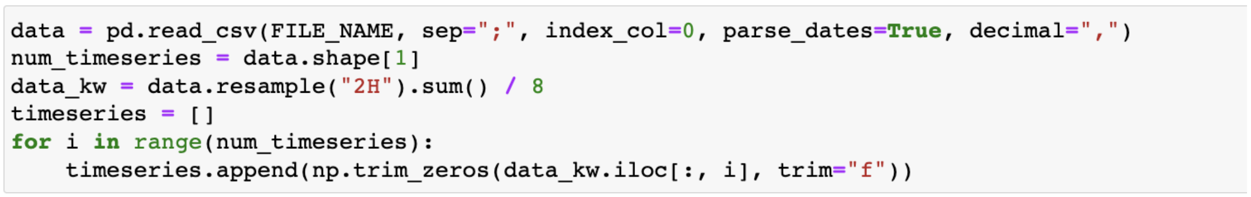
```
s3_bucket = sagemaker.Session().default_bucket()  # replace with an existing bucket if needed
s3_prefix = "deepar-electricity-demo-notebook"  # prefix used for all data stored within the bucket

role = sagemaker.get_execution_role()  # IAM role to use by SageMaker
```

```
region = sagemaker_session.boto_region_name

s3_data_path = "s3://{}/{}/data".format(s3_bucket, s3_prefix)
s3_output_path = "s3://{}/{}/output".format(s3_bucket, s3_prefix)
```

```
image_name = sagemaker.image_uris.retrieve("forecasting-deepar", region)
```

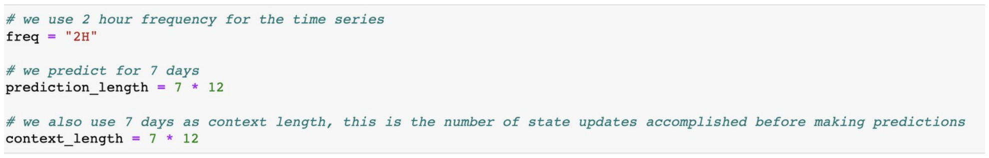
```
DATA_HOST = "sagemaker-sample-files"
DATA_PATH = "datasets/timeseries/uci_electricity/"
ARCHIVE_NAME = "LD2011_2014.txt.zip"
FILE_NAME = ARCHIVE_NAME[:-4]
```

```
s3_client = boto3.client("s3")

if not os.path.isfile(FILE_NAME):
    print("downloading dataset (258MB), can take a few minutes depending on your connection")
    s3_client.download_file(DATA_HOST, DATA_PATH + ARCHIVE_NAME, ARCHIVE_NAME)

    print("\nextracting data archive")
    zip_ref = zipfile.ZipFile(ARCHIVE_NAME, "r")
    zip_ref.extractall("./")
    zip_ref.close()
else:
    print("File found skipping download")
```

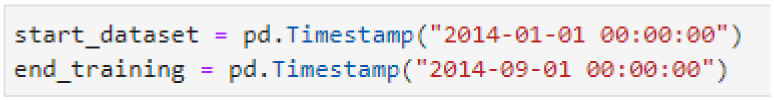


```
data = pd.read_csv(FILE_NAME, sep=";", index_col=0, parse_dates=True, decimal=",")
num_timeseries = data.shape[1]
data_kw = data.resample("2H").sum() / 8
timeseries = []
for i in range(num_timeseries):
    timeseries.append(np.trim_zeros(data_kw.iloc[:, i], trim="f"))
```

B. Type the following in your notebook to train and test splits



```
# we use 2 hour frequency for the time series
freq = "2H"

# we predict for 7 days
prediction_length = 7 * 12

# we also use 7 days as context length, this is the number of state updates accomplished before making predictions
context_length = 7 * 12
```



```
start_dataset = pd.Timestamp("2014-01-01 00:00:00")
end_training = pd.Timestamp("2014-09-01 00:00:00")
```

```python
training_data = [
    {
        "start": str(start_dataset),
        "target": ts[
            start_dataset : end_training - timedelta(days=1)
        ].tolist(),  # We use -1, because pandas indexing includes the upper bound
    }
    for ts in timeseries
]
print(len(training_data))
```

```
370
```

```python
num_test_windows = 4

test_data = [
    {
        "start": str(start_dataset),
        "target": ts[start_dataset : end_training + timedelta(days=k * prediction_length)].tolist(),
    }
    for k in range(1, num_test_windows + 1)
    for ts in timeseries
]
print(len(test_data))
```

```
1480
```

```python
def write_dicts_to_file(path, data):
    with open(path, "wb") as fp:
        for d in data:
            fp.write(json.dumps(d).encode("utf-8"))
            fp.write("\n".encode("utf-8"))
```

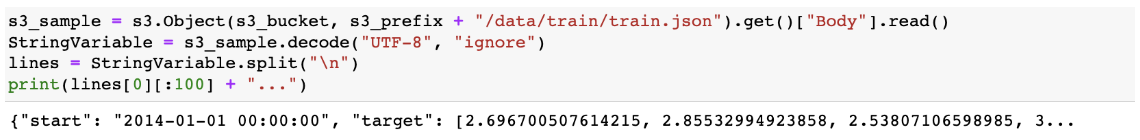
```python
%%time
write_dicts_to_file("train.json", training_data)
write_dicts_to_file("test.json", test_data)
```

```
CPU times: user 4.04 s, sys: 117 ms, total: 4.16 s
Wall time: 4.72 s
```

```python
s3 = boto3.resource("s3")


def copy_to_s3(local_file, s3_path, override=False):
    assert s3_path.startswith("s3://")
    split = s3_path.split("/")
    bucket = split[2]
    path = "/".join(split[3:])
    buk = s3.Bucket(bucket)

    if len(list(buk.objects.filter(Prefix=path))) > 0:
        if not override:
            print(
                "File s3://{}/{} already exists.\nSet override to upload anyway.\n".format(
                    s3_bucket, s3_path
                )
            )
            return
        else:
            print("Overwriting existing file")
    with open(local_file, "rb") as data:
        print("Uploading file to {}".format(s3_path))
        buk.put_object(Key=path, Body=data)
```

```python
%%time
copy_to_s3("train.json", s3_data_path + "/train/train.json")
copy_to_s3("test.json", s3_data_path + "/test/test.json")
```

```
Uploading file to s3://sagemaker-us-east-2-931175847565/deepar-electricity-demo-notebook/data/train/train.json
Uploading file to s3://sagemaker-us-east-2-931175847565/deepar-electricity-demo-notebook/data/test/test.json
CPU times: user 463 ms, sys: 137 ms, total: 600 ms
Wall time: 1.75 s
```

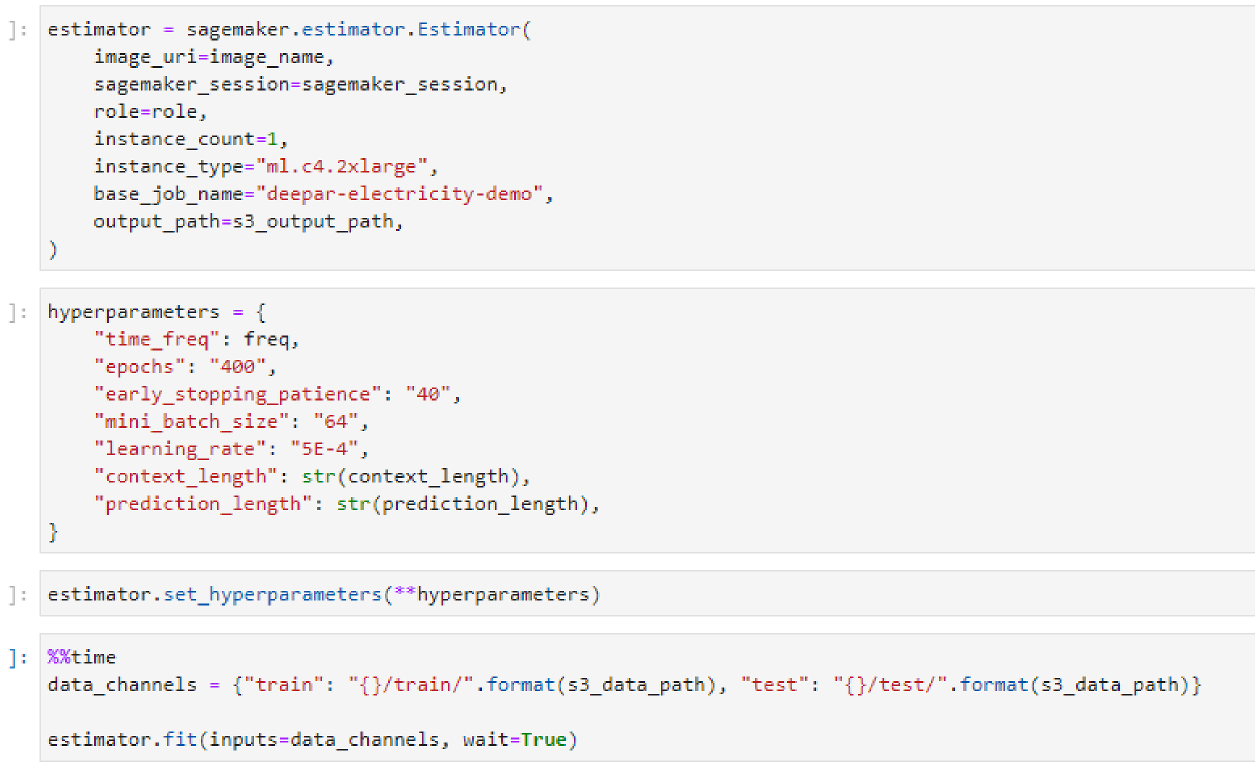


```
s3_sample = s3.Object(s3_bucket, s3_prefix + "/data/train/train.json").get()["Body"].read()
StringVariable = s3_sample.decode("UTF-8", "ignore")
lines = StringVariable.split("\n")
print(lines[0][:100] + "...")
```

```
{"start": "2014-01-01 00:00:00", "target": [2.696700507614215, 2.85532994923858, 2.53807106598985, 3...
```

C. Training the model

Let’s train the DeepAR model once we have the data available in the correct format for training.



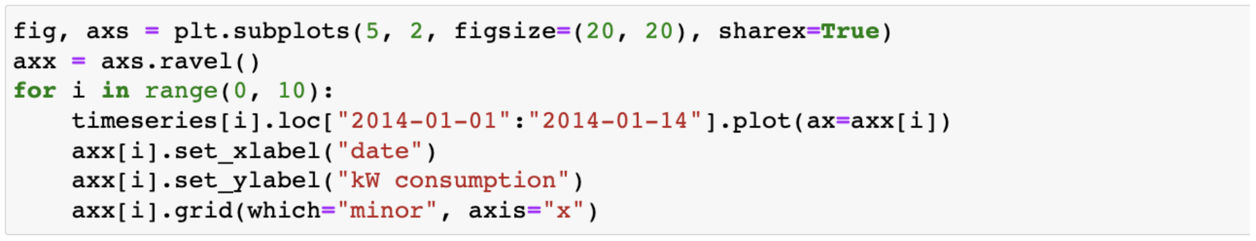
```
]: estimator = sagemaker.estimator.Estimator(
       image_uri=image_name,
       sagemaker_session=sagemaker_session,
       role=role,
       instance_count=1,
       instance_type="ml.c4.2xlarge",
       base_job_name="deepar-electricity-demo",
       output_path=s3_output_path,
   )
```

```
]: hyperparameters = {
       "time_freq": freq,
       "epochs": "400",
       "early_stopping_patience": "40",
       "mini_batch_size": "64",
       "learning_rate": "5E-4",
       "context_length": str(context_length),
       "prediction_length": str(prediction_length),
   }
```

```
]: estimator.set_hyperparameters(**hyperparameters)
```

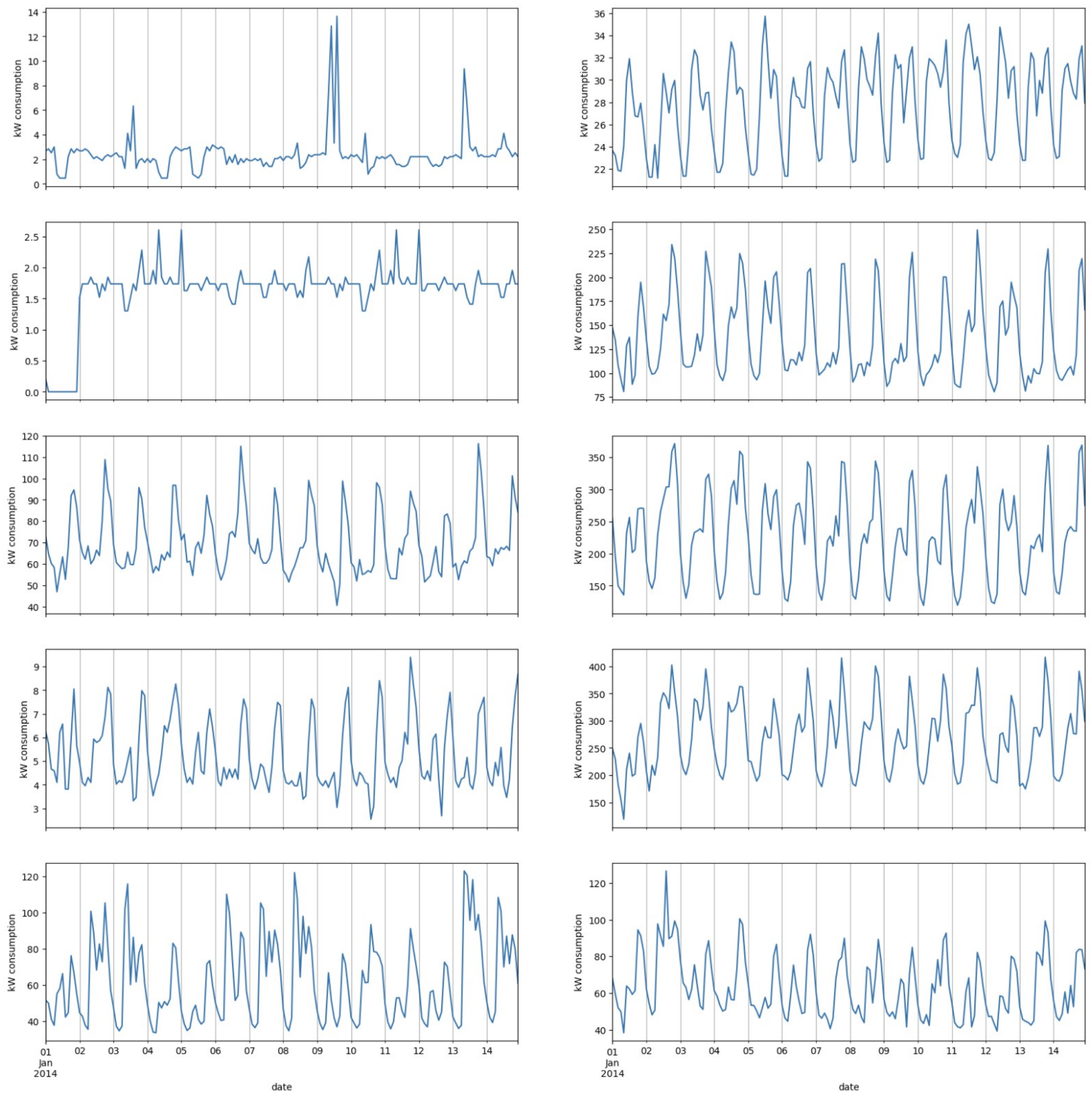
```
]: %%time
   data_channels = {"train": "{}/train/".format(s3_data_path), "test": "{}/test/".format(s3_data_path)}

   estimator.fit(inputs=data_channels, wait=True)
```

Wait for the model to finish training. After that, let’s plot the resulting time series for ten customers for the first two weeks of 2014.



```
fig, axs = plt.subplots(5, 2, figsize=(20, 20), sharex=True)
axx = axs.ravel()
for i in range(0, 10):
    timeseries[i].loc["2014-01-01":"2014-01-14"].plot(ax=axx[i])
    axx[i].set_xlabel("date")
    axx[i].set_ylabel("kW consumption")
    axx[i].grid(which="minor", axis="x")
```

The result is:



Take the screenshot for all steps and save it in your local repository along with your code.

HOS submission instructions:

1. Please install the GitHub Desktop: https://cityuseattle.github.io/docs/git/github_desktop/

2. Clone, organize, and submit your work through GitHub Desktop: https://cityuseattle.github.io/docs/hoporhos