Environment Set-Up

Load relevant Python Packages

In [1]:
```
reset -fs
```

In [2]:
```python
# Importing the most important modules
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import pickle
import time
from matplotlib import pyplot
import matplotlib.dates as mdates
from tqdm.notebook import tqdm

# Import plotly modules to view time series in a more interactive way
import plotly.graph_objects as go
import plotly.offline as pyo
from matplotlib.pyplot import cm
from IPython.display import Image

# Importing time series split for cross validation of time series models
from sklearn.model_selection import TimeSeriesSplit

# For Data Mining
import os, glob
from pandas import read_csv

# For Data Cleaning
from datetime import datetime
import missingno as msno

# Importing metrics to evaluate the implemented models
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
```

Global Variables and Settings

In [3]:
```python
# Setting the random seed for reproducability and several plotting style parameters
%matplotlib inline
plt.style.use('seaborn')
pyo.init_notebook_mode()
sns.set(rc={'figure.figsize':(14,8)})
warnings.filterwarnings('ignore')
pd.set_option('display.max_columns', None)
RSEED = 42
```

Load Data

In [4]:
```python
#data has been saved using a .pkl file.
path = './data/df_small.pkl'
df = pd.read_pickle(path)
df.head(2)
```

Out[4]:

	power_available_mw_obsnorm	target_losses_norm	lagged_NetConsumption_MW	lagged_energyprice_euro_MWh	dswrf_sfc_wm2
2018-01-01 06:00:00	0.891657	0.425598	3142.133333	-71.616667	0.0
2018-01-01 06:10:00	0.911849	0.404513	3144.800000	-72.540000	0.0

Setting Up Training, Validation and Test Dataframes

The dataframe is split into a training set, a validation set (10 consecutive days of data) and a test set (10 consecutive days of data).

In [5]:
```python
#setting up the consecutive test days (last 10 days of dataframe)
test_timestamps = []
for i in range (10):
    test_timestamps.append(pd.to_datetime(df.index[-1]) - (i+1)*pd.Timedelta(hours=24))
test_timestamps.sort()

#setting up the consecutive validation days (10 form March 17 2019 06:00)
val_timestamps = [pd.to_datetime("2019-03-17 06:00:00")]
for i in range (9):
    val_timestamps.append(pd.to_datetime(val_timestamps[0]) + (i+1)*pd.Timedelta(hours=24))
val_timestamps.sort()
```

In [6]:
```python
#splitting dataframe in training, validation and test data
train_df = df[(df.index < val_timestamps[0])]
val_df = df[(df.index >= val_timestamps[0]) & (df.index < val_timestamps[0]+ pd.Timedelta(hours=240))]
test_df = df[(df.index >= test_timestamps[0]) & (df.index < test_timestamps[0]+ pd.Timedelta(hours=240))]
```

General Functions

Error Metrics Function (RMSE, R2, MAE, MAPE)

In [7]:
```python
def error_metrics(y_pred, y_truth, model_name = "default"):
    """
    Calculate error metrics for a single comparison between predicted and observed values
    """
    # calculating error metrics
    RMSE_return = np.sqrt(mean_squared_error(y_truth, y_pred))
    R2_return = r2_score(y_truth, y_pred)
    MAE_return = mean_absolute_error(y_truth, y_pred)
    MAPE_return = (np.mean(np.abs((y_truth - y_pred) / y_truth)) * 100)

    # saving error metrics in a dataframe and returning it
    name_error = ['RMSE', 'R2', 'MAE', 'MAPE']
    value_error = [RMSE_return, R2_return, MAE_return, MAPE_return/100]
    dict_error = dict()
    for i in range(len(name_error)):
        dict_error[name_error[i]] = [value_error[i]]
    errors = pd.DataFrame(dict_error).T
    errors.rename(columns={0 : model_name}, inplace = True)

    #path = './data/error_metrics_{}.pkl'.format(model_name)
    #errors.to_pickle(path)

    return(errors)
```

Naive Base Model - Multi Step Prediction

In [8]:
```python
prediction_steps = 18
```

In [9]:
```python
all_pred_columnnames = list()
all_observed_columnnames = list()
val_errors_columnnames = list()
test_errors_columnnames = list()


for i in range(prediction_steps):
    all_pred_columnnames.append(f"y_all_pred Step {i+1}")
    all_observed_columnnames.append(f"y_all_observed Step {i+1}")
    val_errors_columnnames.append(f"Validation Errors Step {i+1}")
    test_errors_columnnames.append(f"Test Errors Step {i+1}")

y_all_pred = pd.DataFrame(columns = all_pred_columnnames)
y_all_observed = pd.DataFrame(columns = all_observed_columnnames)
val_errors = pd.DataFrame(columns = val_errors_columnnames)
test_errors = pd.DataFrame(columns = test_errors_columnnames)

for i in range(prediction_steps):
    y_all_pred[f"y_all_pred Step {i+1}"] = df["target_losses_norm"].shift(1)
    y_all_observed[f"y_all_observed Step {i+1}"] = df["target_losses_norm"].shift(-(i))

y_all_pred.drop(y_all_pred.head(1).index,inplace=True)
y_all_pred.drop(y_all_pred.tail(18).index,inplace=True)
y_all_observed.drop(y_all_observed.head(1).index,inplace=True)
y_all_observed.drop(y_all_observed.tail(18).index,inplace=True)

y_val_pred = y_all_pred[(y_all_pred.index >= val_timestamps[0]) & (y_all_pred.index < val_timestamps[0]+ pd.Timedelta(hours=240))]
y_val_observed = y_all_observed[(y_all_observed.index >= val_timestamps[0]) & (y_all_observed.index < val_timestamps[0]+ pd.Timedelta(hours=240))]
y_test_pred = y_all_pred[(y_all_pred.index >= test_timestamps[0]) & (y_all_pred.index < test_timestamps[0]+ pd.Timedelta(hours=240))]
y_test_observed = y_all_observed[(y_all_observed.index >= test_timestamps[0]) & (y_all_observed.index < test_timestamps[0]+ pd.Timedelta(hours=240))]

for i in range(prediction_steps):
    val_errors[f"Validation Errors Step {i+1}"]  = error_metrics(y_val_pred[f"y_all_pred Step {i+1}"],y_val_observed[f"y_all_observed Step {i+1}"])["default"]
    test_errors[f"Test Errors Step {i+1}"]  = error_metrics(y_test_pred[f"y_all_pred Step {i+1}"],y_test_observed[f"y_all_observed Step {i+1}"])["default"]

naive_val_errors = val_errors.T
naive_test_errors = test_errors.T
```

In [10]:
```python
y_val_pred["Model"] = "Naive Shift Model"
y_val_observed["Model"] = "Naive Shift Model"
y_test_pred["Model"] = "Naive Shift Model"
y_test_observed["Model"] = "Naive Shift Model"

y_val_pred.to_csv("./Results/naive_shift_validation_predictions.csv", index_label = "date")
y_val_observed.to_csv("./Results/naive_shift_validation_values.csv", index_label = "date")
y_test_pred.to_csv("./Results/naive_shift_test_predictions.csv", index_label = "date")
y_test_observed.to_csv("./Results/naive_shift_test_values.csv", index_label = "date")

print('This cell was last run on: ')
print(datetime.now())
```

```
This cell was last run on:
2020-11-26 10:40:26.151207
```

In [11]:
```
naive_val_errors
```

Out[11]:

	RMSE	R2	MAE	MAPE
Validation Errors Step 1	0.015531	0.990379	0.006981	0.145229
Validation Errors Step 2	0.022869	0.979075	0.010623	0.212613
Validation Errors Step 3	0.027315	0.970042	0.013199	0.267189
Validation Errors Step 4	0.030603	0.962265	0.015377	0.321861
Validation Errors Step 5	0.033607	0.954361	0.017438	0.373985
Validation Errors Step 6	0.036938	0.944826	0.019564	0.429347
Validation Errors Step 7	0.040215	0.934567	0.021507	0.482945
Validation Errors Step 8	0.043191	0.924413	0.023357	0.532200
Validation Errors Step 9	0.046144	0.913604	0.025346	0.578890
Validation Errors Step 10	0.048888	0.902889	0.027107	0.624291
Validation Errors Step 11	0.051248	0.893230	0.028640	0.664189
Validation Errors Step 12	0.052962	0.885929	0.029863	0.700884
Validation Errors Step 13	0.054669	0.878414	0.031212	0.746421
Validation Errors Step 14	0.056456	0.870280	0.032512	0.789113
Validation Errors Step 15	0.058052	0.862773	0.033719	0.839030
Validation Errors Step 16	0.059220	0.857072	0.034659	0.878559
Validation Errors Step 17	0.060112	0.852561	0.035660	0.921963
Validation Errors Step 18	0.061411	0.845966	0.036798	0.966054

In [12]:
```
naive_test_errors
```

Out[12]:

	RMSE	R2	MAE	MAPE
Test Errors Step 1	0.012406	0.994166	0.004951	0.111856
Test Errors Step 2	0.018623	0.986852	0.007549	0.155498
Test Errors Step 3	0.023380	0.979276	0.009846	0.189657
Test Errors Step 4	0.028474	0.969261	0.012122	0.229531
Test Errors Step 5	0.033267	0.958040	0.014356	0.262228
Test Errors Step 6	0.037861	0.945648	0.016536	0.295780
Test Errors Step 7	0.042451	0.931673	0.018565	0.329344
Test Errors Step 8	0.046852	0.916769	0.020602	0.365788
Test Errors Step 9	0.051013	0.901329	0.022409	0.394925
Test Errors Step 10	0.055240	0.884294	0.024192	0.422056
Test Errors Step 11	0.059416	0.866138	0.025977	0.447501
Test Errors Step 12	0.063474	0.847221	0.027811	0.469560
Test Errors Step 13	0.067445	0.827506	0.029529	0.493893
Test Errors Step 14	0.071279	0.807324	0.031213	0.520368
Test Errors Step 15	0.075118	0.785998	0.032898	0.542859
Test Errors Step 16	0.078830	0.764311	0.034554	0.564594
Test Errors Step 17	0.082329	0.742905	0.036110	0.585350
Test Errors Step 18	0.085752	0.721058	0.037626	0.604151

Moving Average - Multi Step Prediction

In [13]:
```python
prediction_steps = 18
span = 2 #averaging over the last 20 minutes
```

In [14]:
```python
all_pred_columnnames = list()
all_observed_columnnames = list()
val_errors_columnnames = list()
test_errors_columnnames = list()


for i in range(prediction_steps):
    all_pred_columnnames.append(f"y_all_pred Step {i+1}")
    val_errors_columnnames.append(f"Validation Errors Step {i+1}")
    test_errors_columnnames.append(f"Test Errors Step {i+1}")

for i in range(prediction_steps+span):
    all_observed_columnnames.append(f"y_all_observed Step {i+1}")


y_all_pred = pd.DataFrame(columns = all_pred_columnnames)
val_errors = pd.DataFrame(columns = val_errors_columnnames)
test_errors = pd.DataFrame(columns = test_errors_columnnames)

y_all_observed = pd.DataFrame(columns = all_observed_columnnames)

for i in range(prediction_steps+span):
    y_all_observed[f"y_all_observed Step {i+1}"] = df["target_losses_norm"].shift(-i)



y_pred_step1 = list()

for index, row in y_all_observed.iterrows():
    y_pred_step1.append((pd.Series([row['y_all_observed Step 1'], row['y_all_observed Step 2']]).rolling(window=span, min_periods=span).mean().iloc[1]))

y_all_pred["y_all_pred Step 1"] = pd.Series(y_pred_step1)
y_all_pred.index = y_all_observed.index

y_pred_step2 = list()

for index, row in y_all_observed.iterrows():
    y_pred_step2.append((pd.Series([row['y_all_observed Step 2'], y_all_pred["y_all_pred Step 1"].loc[index]]).rolling(window=span, min_periods=span).mean().iloc[1]))

y_all_pred["y_all_pred Step 2"] = pd.Series(y_pred_step2, index = y_all_observed.index)

for i in range(2,prediction_steps+span):
    storage_list = list()
    for index, row in y_all_pred.iterrows():
        storage_list.append((pd.Series([row[f'y_all_pred Step {i-1}'], row[f'y_all_pred Step {i}']]).rolling(window=span, min_periods=span).mean().iloc[1]))
    y_all_pred[f"y_all_pred Step {i+1}"] = pd.Series(storage_list, index = y_all_observed.index)
```

In [15]:
```python
y_all_pred.drop(y_all_pred.head(1).index,inplace=True)
y_all_pred.drop(y_all_pred.tail(19).index,inplace=True)
y_all_observed.drop(y_all_observed.head(1).index,inplace=True)
y_all_observed.drop(y_all_observed.tail(19).index,inplace=True)

y_val_pred = y_all_pred[(y_all_pred.index >= val_timestamps[0]) & (y_all_pred.index < val_timestamps[0]+ pd.Timedelta(hours=240))]
y_val_observed = y_all_observed[(y_all_observed.index >= val_timestamps[0]) & (y_all_observed.index < val_timestamps[0]+ pd.Timedelta(hours=240))]
y_test_pred = y_all_pred[(y_all_pred.index >= test_timestamps[0]) & (y_all_pred.index < test_timestamps[0]+ pd.Timedelta(hours=240))]
y_test_observed = y_all_observed[(y_all_observed.index >= test_timestamps[0]) & (y_all_observed.index < test_timestamps[0]+ pd.Timedelta(hours=240))]

for i in range(prediction_steps):
    val_errors[f"Validation Errors Step {i+1}"]  = error_metrics(y_val_pred[f"y_all_pred Step {i+1}"],y_val_observed[f"y_all_observed Step {i+3}"])["default"]
    test_errors[f"Test Errors Step {i+1}"]  = error_metrics(y_test_pred[f"y_all_pred Step {i+1}"],y_test_observed[f"y_all_observed Step {i+3}"])["default"]

mov_av_val_errors = val_errors.T
mov_av_test_errors = test_errors.T
```

In [16]:
```python
y_val_pred["Model"] = "Moving Average Model"
y_val_observed["Model"] = "Moving Average Model"
y_test_pred["Model"] = "Moving Average Model"
y_test_observed["Model"] = "Moving Average Model"

y_val_pred.to_csv("./Results/moving_average_validation_predictions.csv", index_label = "date")
y_val_observed.to_csv("./Results/moving_average_validation_values.csv", index_label = "date")
y_test_pred.to_csv("./Results/moving_average_test_predictions.csv", index_label = "date")
y_test_observed.to_csv("./Results/moving_average_test_values.csv", index_label = "date")

print('This cell was last run on: ')
print(datetime.now())
```

```
This cell was last run on:
2020-11-26 10:57:41.306077
```

In [17]:
```
mov_av_val_errors
```

Out[17]:

	RMSE	R2	MAE	MAPE
Validation Errors Step 1	0.017652	0.987489	0.008212	0.166773
Validation Errors Step 2	0.022829	0.979002	0.010748	0.216605
Validation Errors Step 3	0.027418	0.969622	0.013444	0.277849
Validation Errors Step 4	0.030459	0.962483	0.015421	0.329172
Validation Errors Step 5	0.033772	0.953854	0.017621	0.383681
Validation Errors Step 6	0.037279	0.943689	0.019792	0.439536
Validation Errors Step 7	0.040534	0.933337	0.021752	0.491739
Validation Errors Step 8	0.043547	0.922948	0.023584	0.540011
Validation Errors Step 9	0.046342	0.912693	0.025461	0.587568
Validation Errors Step 10	0.048932	0.902628	0.027107	0.630349
Validation Errors Step 11	0.051082	0.893848	0.028512	0.669467
Validation Errors Step 12	0.052841	0.886359	0.029803	0.708322
Validation Errors Step 13	0.054623	0.878505	0.031204	0.754483
Validation Errors Step 14	0.056425	0.870243	0.032553	0.800404
Validation Errors Step 15	0.057934	0.863053	0.033688	0.845920
Validation Errors Step 16	0.059004	0.857805	0.034638	0.887649
Validation Errors Step 17	0.059989	0.852932	0.035680	0.931691
Validation Errors Step 18	0.061305	0.846284	0.036828	0.977387

In [18]:
```
mov_av_test_errors
```

Out[18]:

	RMSE	R2	MAE	MAPE
Test Errors Step 1	0.014561	0.991966	0.005878	0.123604
Test Errors Step 2	0.019188	0.986049	0.007830	0.156641
Test Errors Step 3	0.024699	0.976883	0.010431	0.196628
Test Errors Step 4	0.029508	0.967004	0.012612	0.233208
Test Errors Step 5	0.034422	0.955100	0.014909	0.267468
Test Errors Step 6	0.039003	0.942353	0.016984	0.300662
Test Errors Step 7	0.043600	0.927964	0.019087	0.336358
Test Errors Step 8	0.047934	0.912926	0.021050	0.369660
Test Errors Step 9	0.052152	0.896924	0.022869	0.398673
Test Errors Step 10	0.056382	0.879523	0.024676	0.425942
Test Errors Step 11	0.060539	0.861099	0.026501	0.450948
Test Errors Step 12	0.064583	0.841914	0.028324	0.473500
Test Errors Step 13	0.068522	0.822032	0.030022	0.498777
Test Errors Step 14	0.072371	0.801464	0.031745	0.524636
Test Errors Step 15	0.076178	0.780009	0.033412	0.546087
Test Errors Step 16	0.079827	0.758407	0.035048	0.567377
Test Errors Step 17	0.083310	0.736848	0.036581	0.587307
Test Errors Step 18	0.086700	0.714971	0.038082	0.604090

In [19]:
```python
naive_val_errors.to_csv("./Validation Errors/validation_errors_naive_model.csv", index_label = "Step")
naive_test_errors.to_csv("./Test Errors/test_errors_naive_model.csv", index_label = "Step")
mov_av_val_errors.to_csv("./Validation Errors/validation_errors_moving_average_model.csv", index_label = "Step")
mov_av_test_errors.to_csv("./Test Errors/test_errors_moving_average_model.csv", index_label = "Step")

print('This cell was last run on: ')
print(datetime.now())
```

```
This cell was last run on:
2020-11-26 10:57:41.438104
```