WESTERN POWER DISTRIBUTION

EFFS

ELECTRICITY FLEXIBILITY AND FORECASTING SYSTEM

Forecasting Dissemination Webinar

27 June 2019

RIIO NIC NETWORK INNOVATION COMPETITION

Contents

Project Background

Project aims:

Develop and demonstrate optimal forecasting arrangements for different time-horizons to support the EFFS project

Forecast requirements:

Load and generation at a range of locations and timescales (several months ahead to intraday) as an essential input to power flow analysis

Project requirements:

Develop a reliable forecasting system that supports the integration and evaluation of multiple methods

Methods based on open source technology that can be reproduced and implemented by DNOs

Learning report to be shared with the industry

Project Background

SGS methodology and approach for forecasting:

Implement a **toolchain from open source tools**, segregate the datasets and evaluate multiple methods:

- Auto Regressive Moving Average (ARIMA)
- Long Short Term Memory (LSTM) Artificial Neural Network
- **Extreme Gradient Boost (XGBoost)**

An agile delivery methodology to 'fail fast'. Methods developed and tested by Smarter Grid Solutions ('SGS')

Capita Validation Testing

The SGS methodology using XGBoost was applied by Capita on a broader sample of WPD network locations in order to validate and compare results

Forecasting Methodology

General Approach to Developing Forecasting Models:

Problem Definition → Explore the Data → Model Selection → Evaluate Model Performance

Problem Definition, Explore the Data: Selection of use cases and identification of available data

Model Selection, Evaluate Model Performance: Build models using different methods/data, test and refine

Forecasting Methodology

Initial Use Cases (Problem Definitions):

1. **UC1** – 6 months ahead, GSP study – forecasts for the subsequent 6 months will be provided in 30min time steps.
2. **UC2** – 1 month ahead, BSP study – forecasts for the following month will be provided in 30min time steps.
3. **UC3** – Day ahead, primary study – forecasts for the next 24h in 30min time steps.
4. **UC4** – Hour ahead, BSP study – forecasts for the next 2 half hours.

Forecasting Methodology

Using Agile to select methods:

- UC 1: BSP
- UC 2: GSP
- UC 3: Primary
- UC 4: BSP

- Select data/training parameters
- Test & refine
- Eliminate methods

- ARIMA
- LSTM ANN
- XGBoost

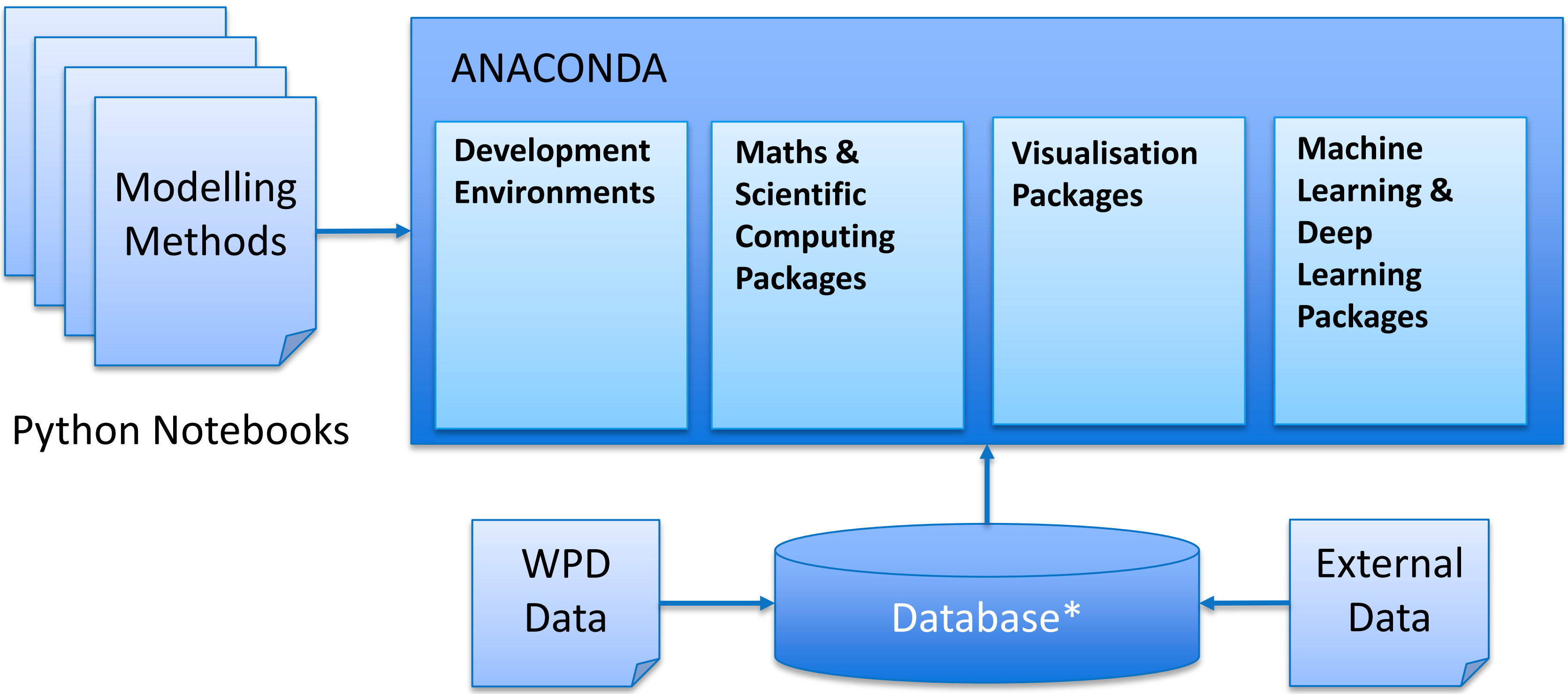
XGBoost

- UC 1: BSP
- UC 2: GSP
- UC 3: Primary
- UC 4: BSP
- UC 5: Primary
- UC 6: Wind
- UC 7: Solar
- UC 8: Large Load

- Select data/training parameters
- Test & refine

Forecasting Methodology

Open Source Tool Chain:



* Please refer to the *Forecasting Methods for EFFS* report for further information on the set-up of a PostgreSQL database and TimescaleDB extension

Results Obtained

- **Model performance:** for the majority of cases XGBoost outperformed the other methods*
- **Forecasting at different voltage levels and substation types**:
  - Techniques based on historical data work best on short time horizons. Observed across all levels.
  - For the Primary and BSP cases with low penetration of wind and solar, relative to yearly demand, a feature set containing only temporal trends will provide predictions with acceptable levels of accuracy; for higher penetrations of renewables, predictions benefit from the addition of weather features to meet accuracy requirements.
  - **GSPs with a mix of DER and high capacity levels, compared to demand levels,** are difficult to forecast (Cardiff versus Truro). Creating an aggregate forecast seems to work better.

* In the majority of cases tested, XGBoost proved faster to tune and train as well as providing accuracies that were in line with or better than other models. In addition, XGBoost proved more convenient for applications across different locations in the WPD network and can be applied to data from other DNOs

Results Obtained

Equilibrium project accuracy (%) results

Category	>50% Accuracy	>80% Accuracy
Primary	84.22	72.55
BSP	26.45	3.46
Wind Generation	39.95	17.37

Use Case	Accuracy	Time Horizon: Six Months Ahead	Month Ahead	Week Ahead	Day Ahead	Hour Ahead
UC1 – GSP	>50%	30.61	28.89	25.07	30.95	50.00
	>80%	11.91	11.69	9.42	13.39	25.00
UC2 – BSP	>50%	99.42	99.94	99.78	100.00	100.00
	>80%	79.23	83.50	92.11	97.32	100.00
UC3 – Primary	>50%	98.23	99.98	100.00	100.00	100.00
	>80%	96.05	98.59	99.33	99.70	100.00
UC4 – BSP	>50%	68.99	73.48	73.41	85.12	100.00
	>80%	29.88	33.75	34.10	45.54	52.08
UC5 – Primary	>50%	97.54	97.74	98.96	100.00	100.00
	>80%	87.36	86.97	91.39	98.51	100.00
UC6 – Wind Generation	>50%	37.33	40.35	48.91	87.20	87.50
	>80%	12.76	18.68	27.49	71.73	79.17
UC7 Solar Generation	>50%	72.28	73.08	77.38	76.19	89.58
	>80%	58.16	54.70	52.68	60.12	62.50
UC8 – Large	>50%	N/A	66.66	71.58	79.17	100.00
	>80%	N/A	27.43	29.41	47.32	93.75

Please see slide 21 for comparison of SGS results with Capita’s validation testing

Results Obtained

Benchmarking with KASM Project

The KASM project also assessed the accuracy of its proprietary ensemble forecasting method but using different metrics. The EFFS results compare favourably when looking at the MAPE and RSME/Capacity figures achieved:

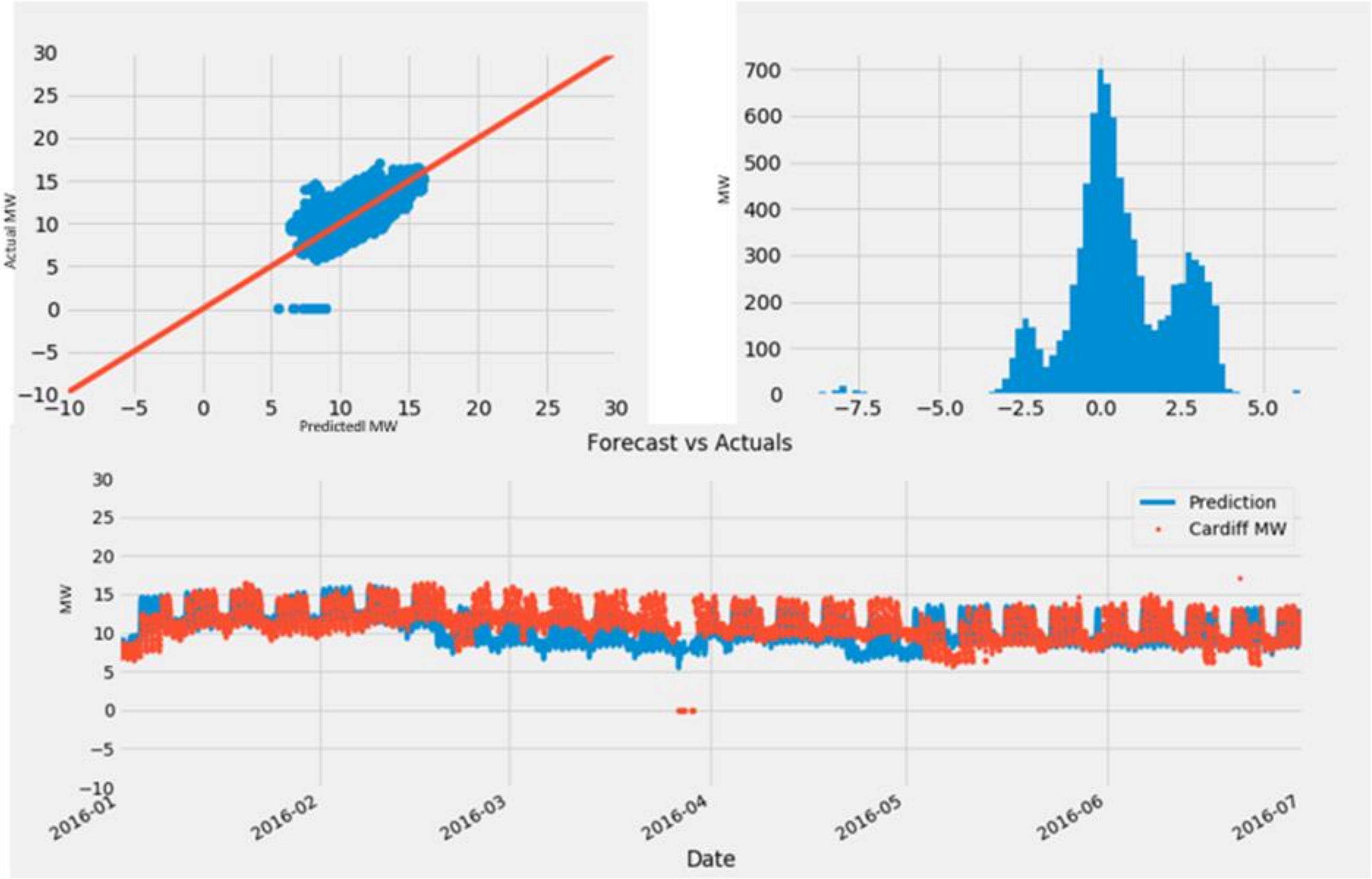
- **MAPE for Load:**
  - KASM: 9% day ahead approximation
  - EFFS: 3.5% day ahead average
- **RMSE/Capacity for Solar:**
  - KASM: 10% day ahead approximation
  - EFFS: 8.4% day ahead average
- **RMSE/Capacity for Wind:**
  - KASM: 16% day ahead approximation
  - EFFS: 12.5% day ahead average

Glossary of terms:

KASM Project	Kent Active System Management project performed by UK Power Networks
MAPE	Mean Absolute Percentage Error
RMSE	Root Mean Squared Error

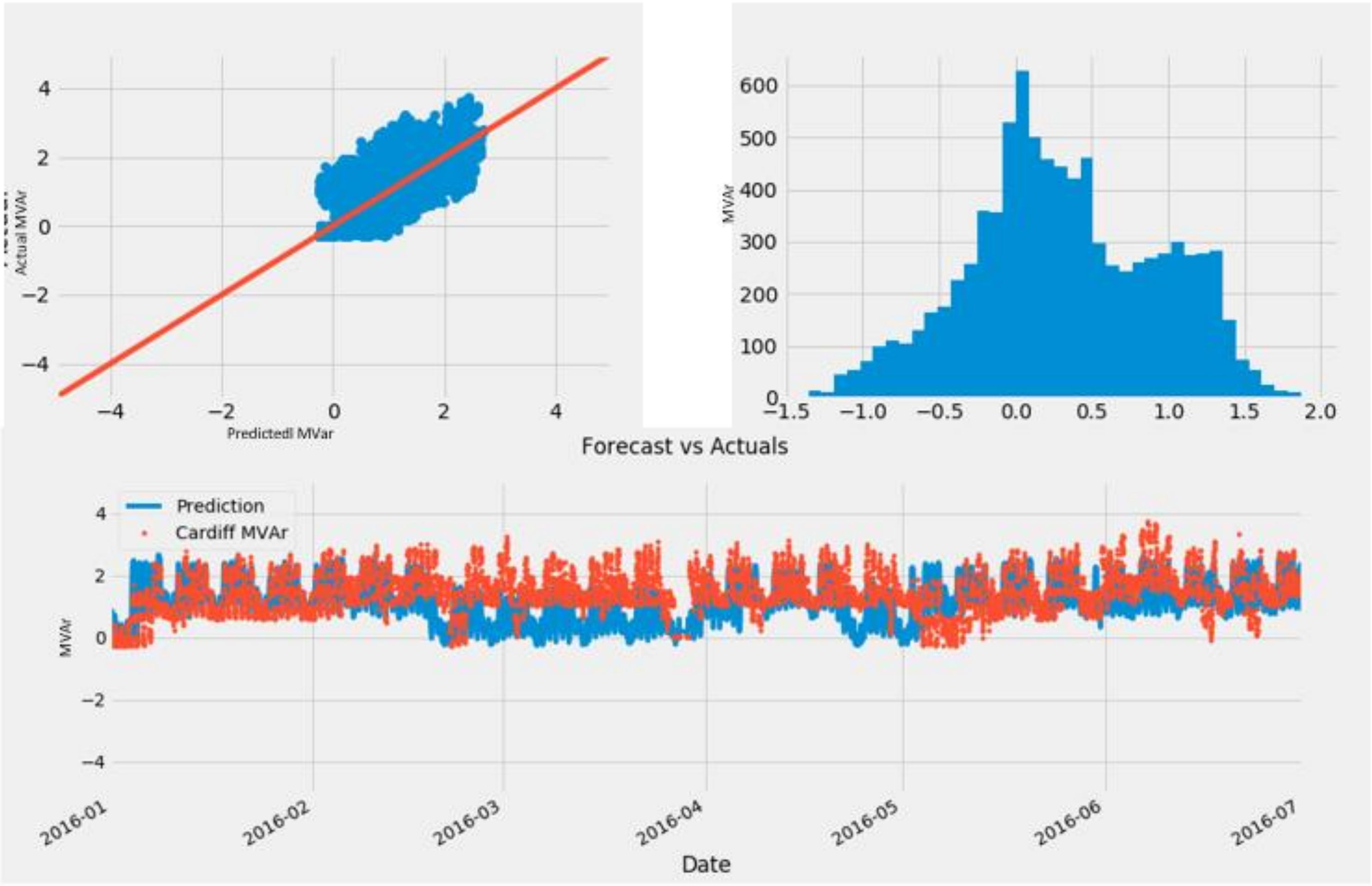
Results Obtained

Cardiff South – 6 months ahead MW



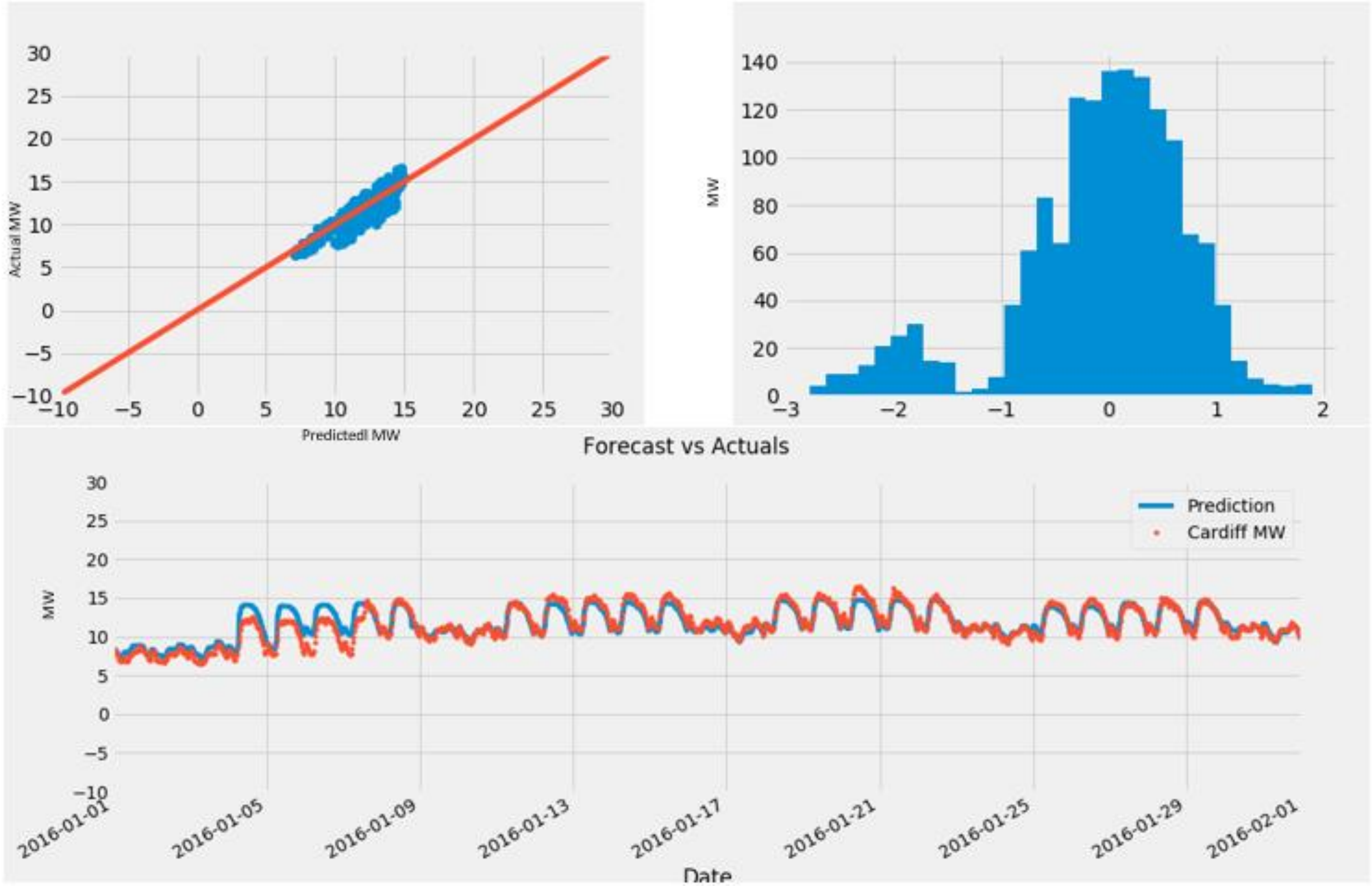
Results Obtained

Cardiff South – 6 months ahead MVAr



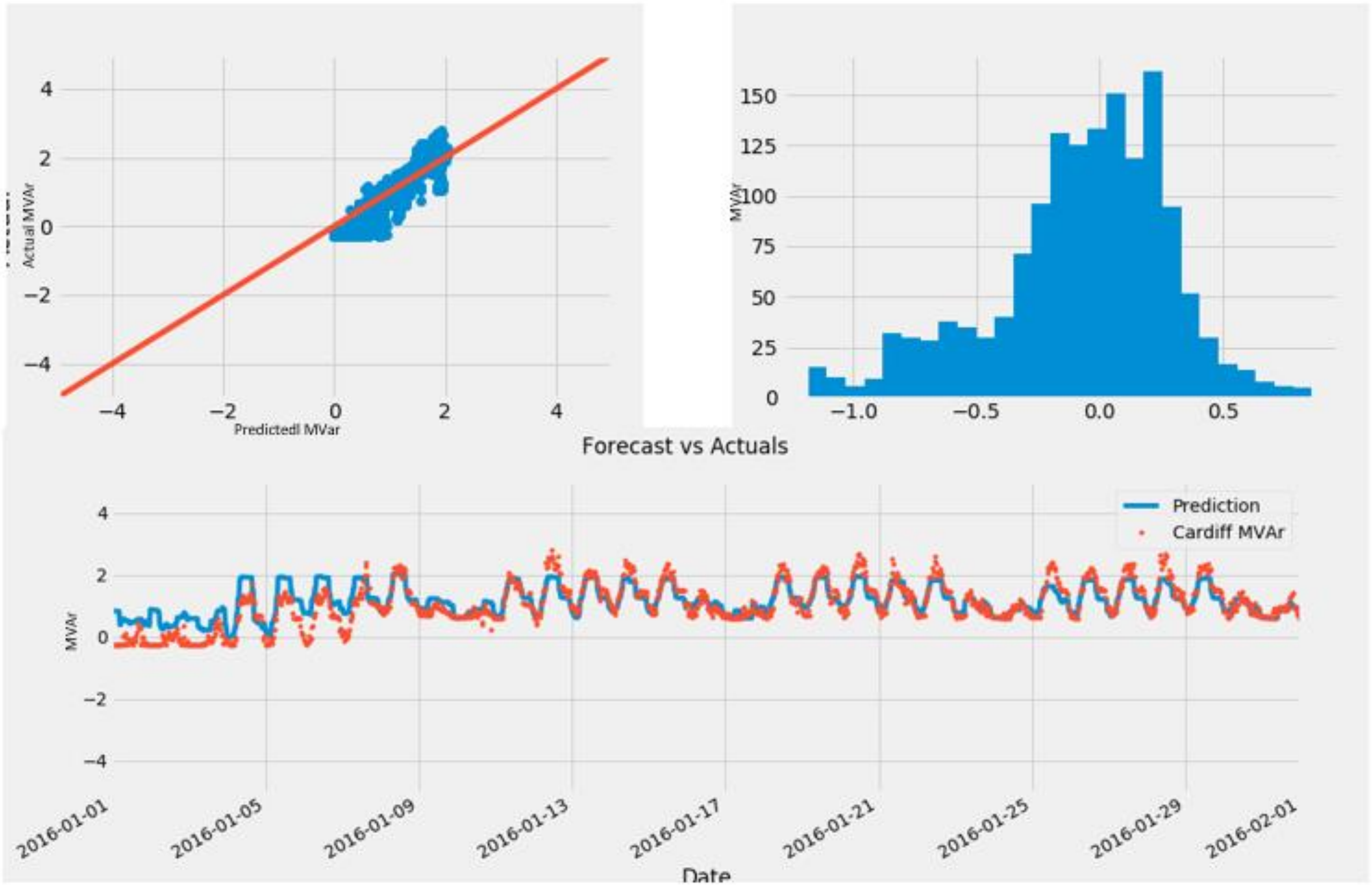
Results Obtained

Cardiff South – 1 month ahead MW



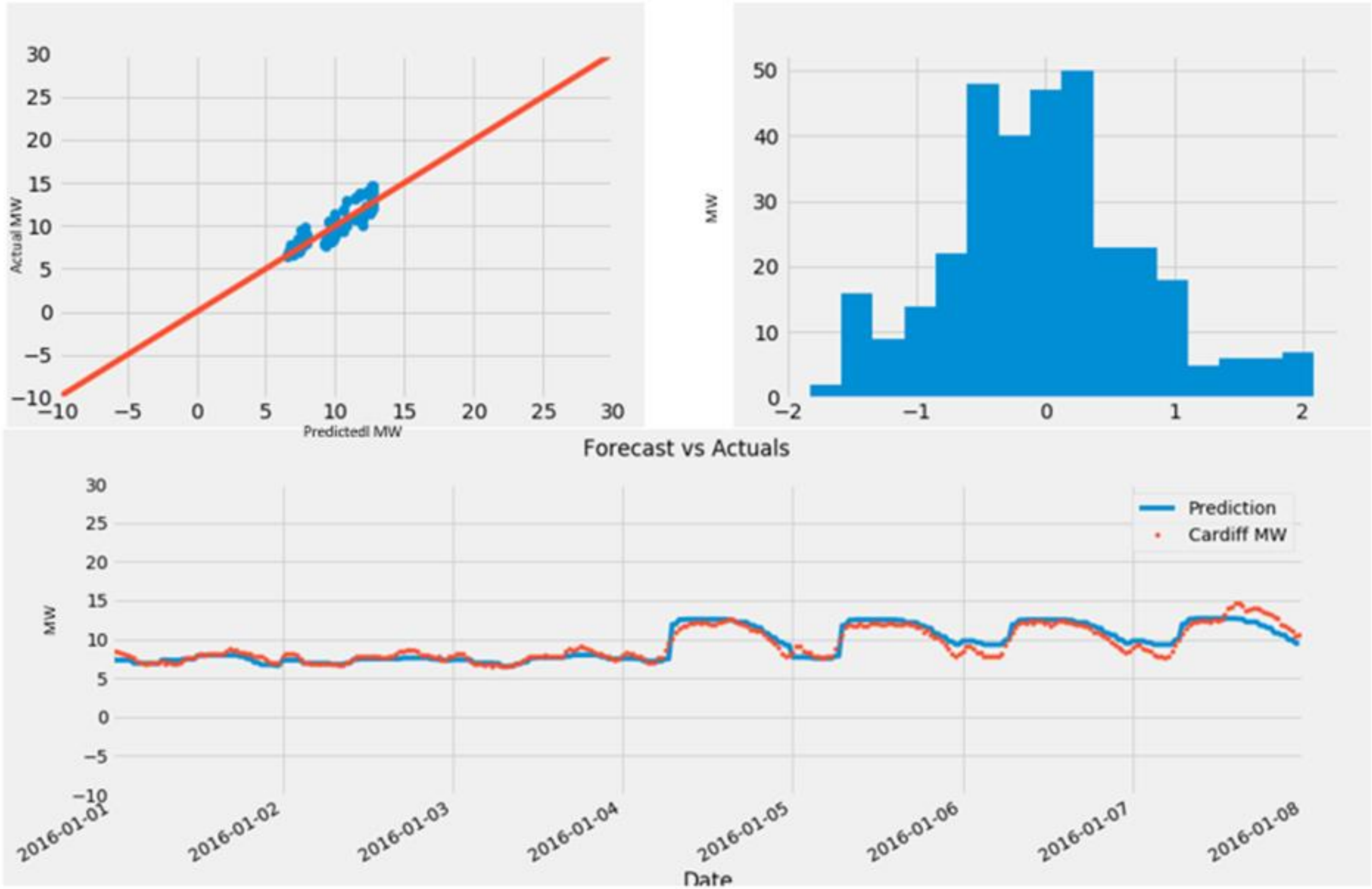
Results Obtained

Cardiff South – 1 month ahead MVAr



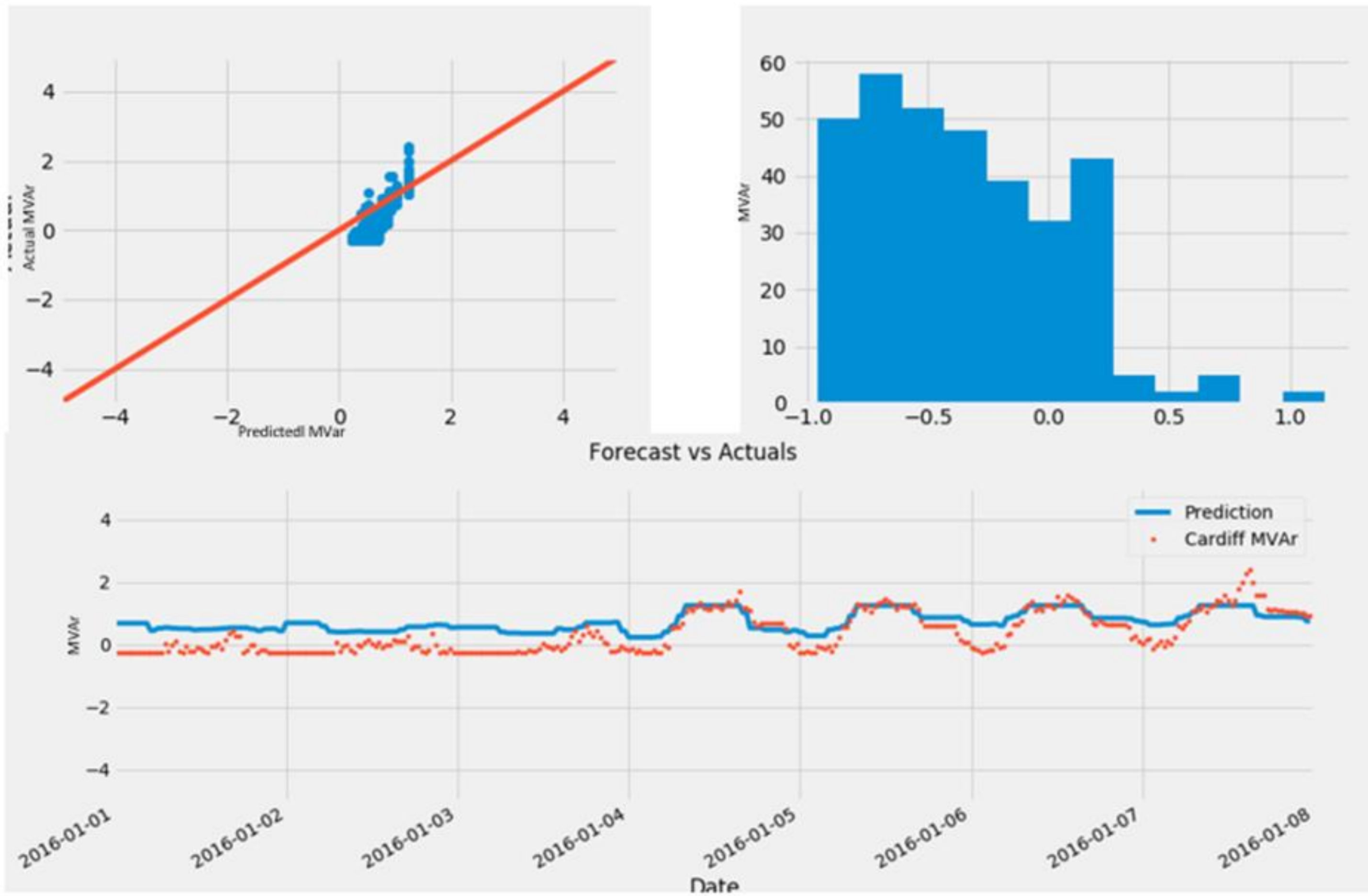
Results Obtained

Cardiff South – week ahead MW



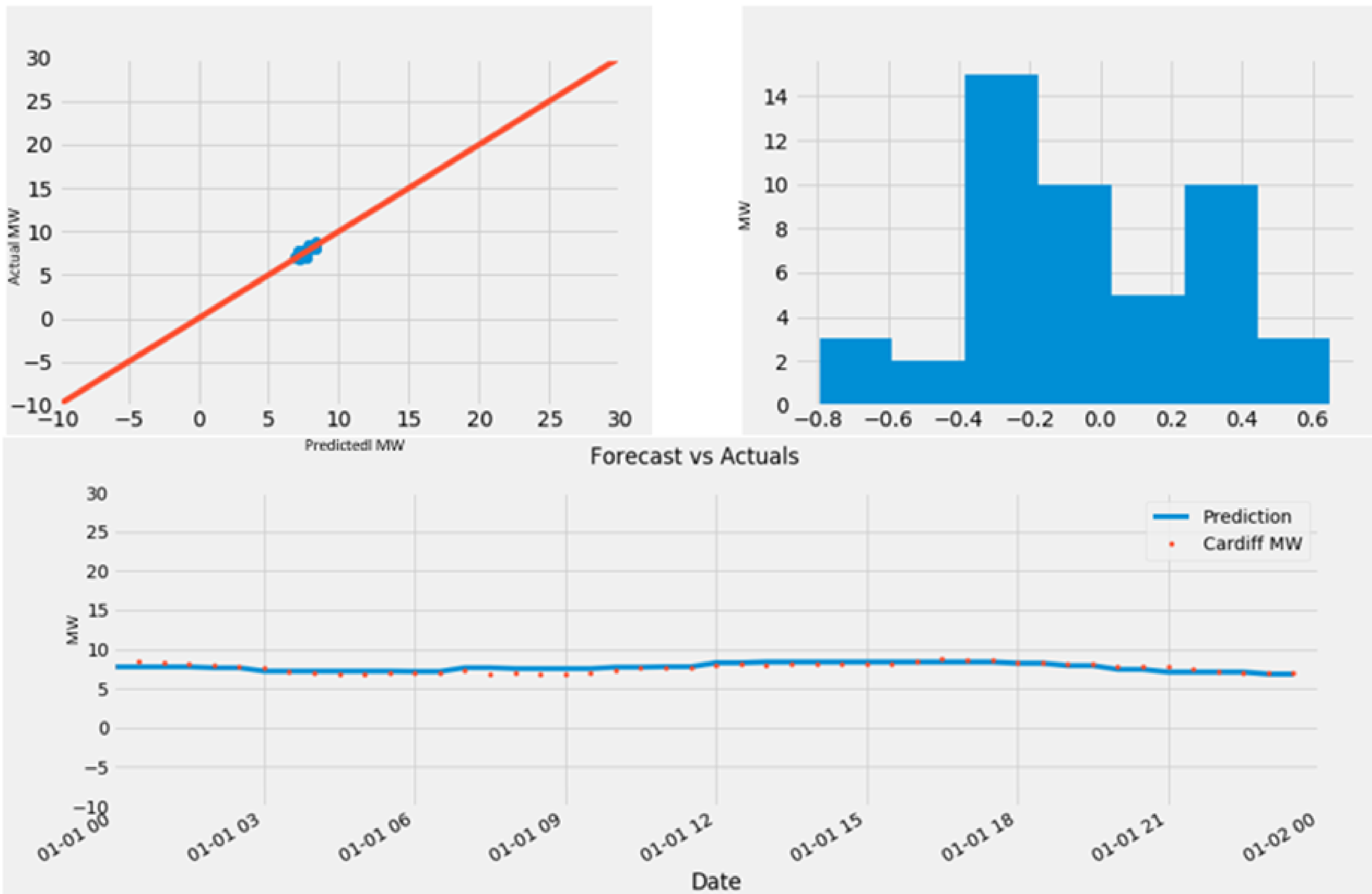
Results Obtained

Cardiff South – week ahead MVar



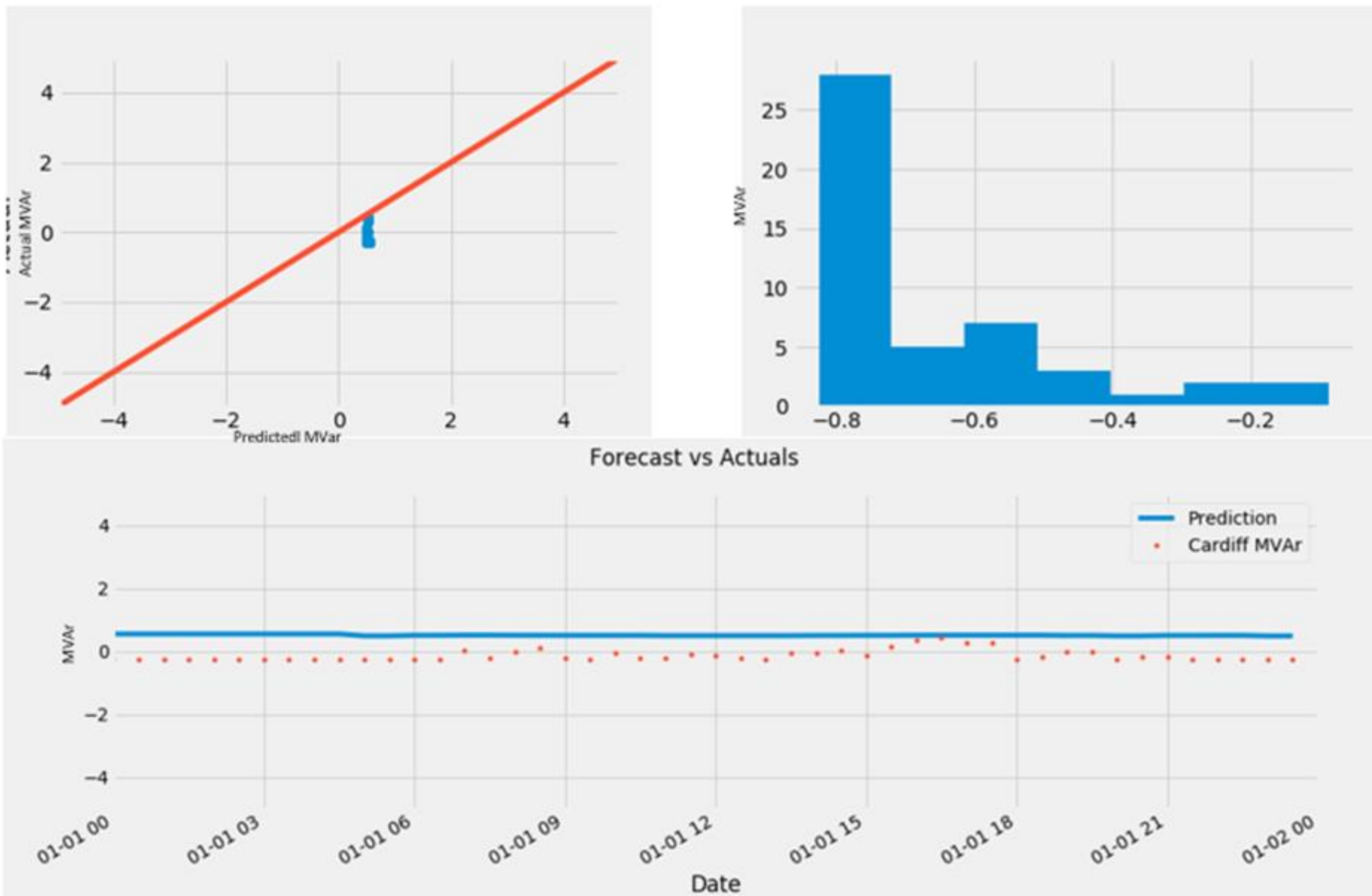
Results Obtained

Cardiff South – day ahead MW



Results Obtained

Cardiff South – day ahead MVAr



CAPITA

Validation Testing – Overview

Aim of validation testing:

- Replication of environment and results by SGS on the same use cases
- Real-world forecasting simulation using SGS model on a broader sample of locations, including GSPs, BSPs, Primaries, load customers and generation customers*
- Testing across multiple time-splits for each selected location and time horizon
  - 6 simulations for 6-month forecasts
  - 20 simulations for all other time horizons

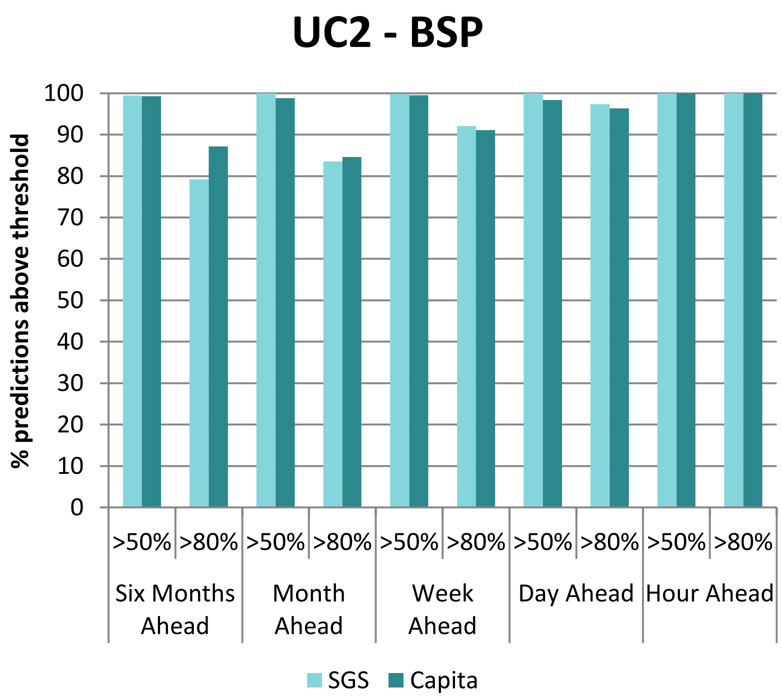
Questions to be answered:

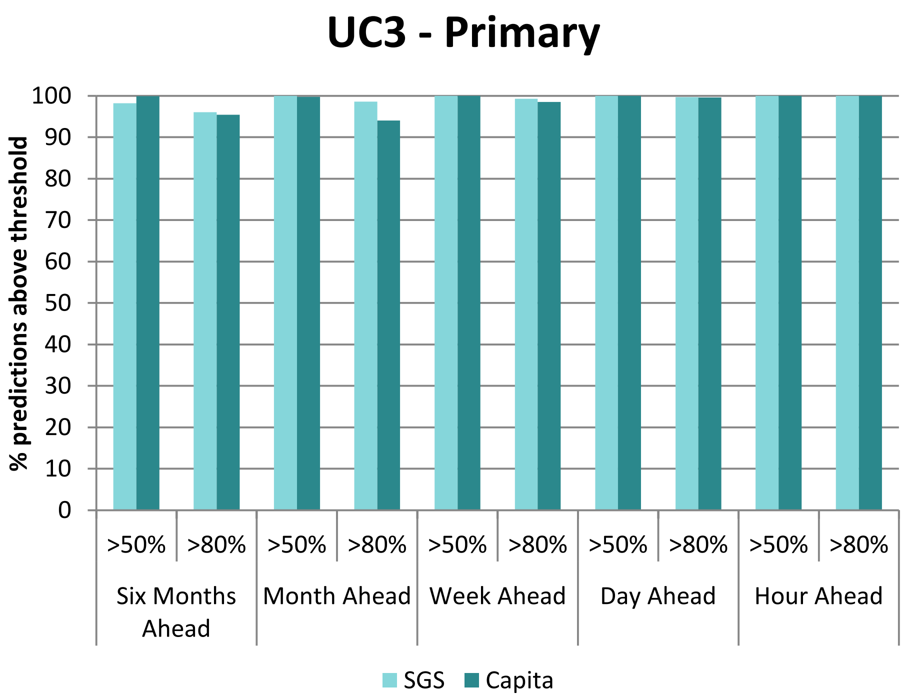
- Consistency of results over time for a given location
- Consistency of results across locations of the same type
- Comparison of results between types of location and time horizons

* Results for BSPs and Primaries are presented, testing for GSPs, generation and load customers still in progress

Validation Testing – Comparison vs. SGS

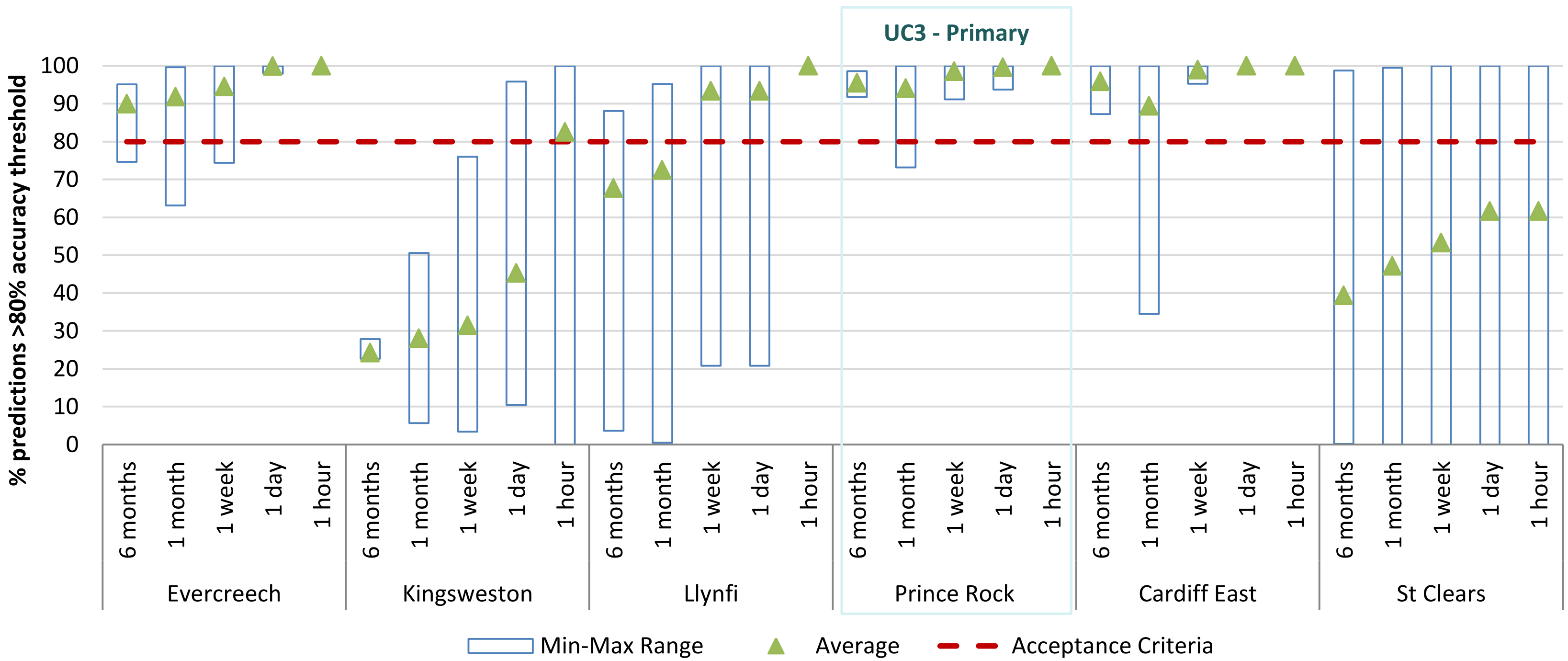
- Successfully replicated results achieved by SGS for the same combination of: location, time horizon, time split, model and model parameters
- For the use cases explored by SGS, Capita's results (averaged over multiple time-splits) show a close match where accuracies are high (e.g. UC2 and UC3)





Validation Testing - Primaries

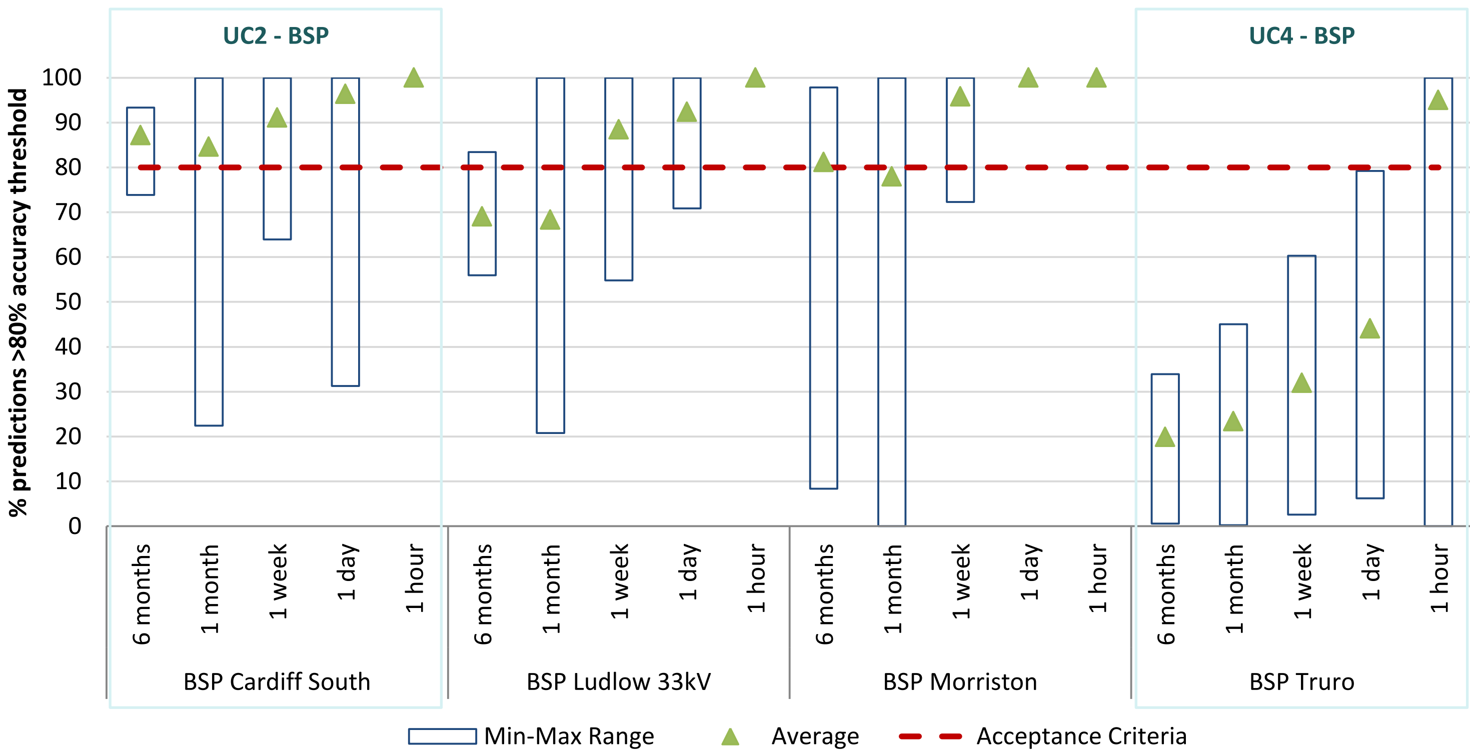
- Chart shows the frequency of passing acceptance criteria for the range of simulations performed: average pass rate, best-worst range and the 80% threshold
  - Evercreech, Prince Rock and Cardiff East are on average above the acceptance criteria
  - Llynfi has good quality data, but the behaviour is less predictable
  - Kingsweston and St Clears affected by data quality issues



CAPITA

Validation Testing - BSPs

- 3 of 4 BSPs tested pass the acceptance criteria for shorter time horizons (up to 1 week)
- Accuracy increases for shorter time horizons
- BSP Truro affected by data quality issues



CAPITA

Validation Testing – Conclusions

High-level conclusions:

- Model performs well for BSPs and Primaries where input data is of good quality
- Accuracy increases with shorter time horizons
- Results that do not meet set thresholds are usually due to data quality issues
- Certain locations exhibit less predictable behaviour, leads to variations in the quality of forecasts

Recommendations for transition to BAU:

- Performing a data quality check relating to each location
- Some locations work ‘out of the box’, others may require additional features (e.g. weather data) to provide reliable forecasts
- Location’s underlying behaviour is key to understanding the robustness of forecasts
- Models developed by SGS represent a starting point for DNOs to adopt forecasting into BAU

Technical Environment

- Based on Anaconda, all methods and libraries are open source
- The environment and methods are described in the *Forecasting Methods for EFFS* report in detail to allow other DNOs and stakeholder to test the methods on their own data. This includes:
  - Instructions on setting up the environment
  - Database schema and set-up instructions
  - Python code (Jupyter notebooks in Anaconda environment)

Technical Environment

```
dateparse = lambda dates: pd.datetime.strptime(dates, '%d/%m/%Y %H:%M')
df = pd.read_csv('C:\\filepath\\input_data.csv',index_col=0,date_parser=dateparse)
#df.index=pd.to_datetime(df['Date'])
print(df.head(10))
print(df.dtypes)

split_date = '2015-06-23'
train = df.loc[df.index <= split_date].copy()
test = df.loc[df.index > split_date].copy()

def create_features(df, label=None):
    """
    Creates time series features from datetime index
    """
    df['date'] = df.index
    df['hour'] = df['date'].dt.hour
    df['dayofweek'] = df['date'].dt.dayofweek
    df['quarter'] = df['date'].dt.quarter
    df['month'] = df['date'].dt.month
    df['year'] = df['date'].dt.year
    df['dayofyear'] = df['date'].dt.dayofyear
    df['dayofmonth'] = df['date'].dt.day
    df['weekofyear'] = df['date'].dt.weekofyear
    #df['Temp'] = df['Temp']

    X = df[['hour','dayofweek','quarter','month','year',
           'dayofyear','dayofmonth','weekofyear']]
    if label:
        y = df[label]
        return X, y
    return X

train, y_train = create_features(train, label='Load')
valid, y_valid = create_features(test, label='Load')
```

Q&A