Power & Utilities (P&U) Forecasting Framework (early access release v0.1)

1. Overview

Electrification and decarbonization via deep penetration of renewables (solar, wind, hydro etc) supply on the electric grid are two major trends reshaping the landscape, and contributing to combating climate change. However, integrating renewables is not straightforward as these sources are inherently intermittent due to their dependence on prevalent microweather conditions which are highly volatile and non-stationary (i.e. hard to forecast). Moreover, unlike renewable-friendly contracts of the past, the new-age energy contracts with a third-party energy buyer, such as Round-The-Clock (RTC), enforce commitment of a prerequisite amount of energy. Therefore, over time, uncertainty in the generation will potentially lead to lower price realization that impedes renewables investments. Thus, accurate renewable generation forecasting helps choose a series of optimal decisions under uncertainty while bidding in the energy markets for better monetization. Renewable farm operators, Power Utilities and other energy traders need to build several new cloud-based advanced AI solutions to support the clean energy transition.

Customers and ISVs/SIs need help in managing the life-cycle of advanced AI-driven industry solutions for Power & Utilities. Small improvements in underlying AI solutions (or plumbing) can lead to significant improvement in business outcomes; but solutions should be stable and should not need to be significantly re-engineered if underlying components are updated to take advantage of latest innovations in AI technologies. Also, industry domain data scientists may choose to specialize a lot more on industry knowledge and have basic data science & engineering skills.

To support these industry needs, Microsoft Cloud for Industry (MCI) has developed Power & Utility Industry-specific AI accelerator IP to simplify the creation, evolution and lifecycle management of P&U AI solutions. Specifically, micro-weather, power (demand, supply) and energy market price forecasting is a journey involving fusion of on-site IoT data inputs (that grow with time), off-site data and exogeneous forecast feature inputs (from multiple data ISV services), and P&U industry specialized pre-processing (exploratory data analysis) and post-processing (go-live, data quality surveillance, AI model maintenance and update) needs. In this early access release (v0.1), Microsoft has captured a set of industry patterns, abstractions, forecasting models and pre-/post-processing tools as part of a framework aimed at P&U domain data scientists and engineers.

In this early access release (v0.1), we have built the first version of an extensible, scalable, broadly applicable P&U forecasting framework that provides a ready implementation for a suite of baseline algorithms along with state-of-the-art advanced forecasting algorithms developed in-house through a unified interface built on Azure Machine Learning (AML). In **Figure 1**, we describe the complete lifecycle along with the value proposition provided by our forecasting framework. This is aimed to help various domain data solutioning stakeholders (Data Engineer, Data Scientist, ML Engineer, Dev Engineer) to shrink the innovation cycle by reducing the boilerplate coding effort and providing a standard problem definition structure along with support for versioning, logging, and continuous monitoring of deployed models.

Below, we list industry-specific abstractions provided over the entire forecasting lifecycle -

- We provide a suite of Exploratory Data Analysis (EDA) that affects solar, wind, price, and demand forecasting requirements.
- We formalize the pattern defining the time series forecasting algorithms relevant to the P&U context. Specifically, we handle exogenous variables (both pas and future covariates) for appropriate dta alignment for model training.
- We provide a wrapper that allows for flexible instantiation of DeepMC --- an in-house algorithm designed for micro-climate prediction --- to forecasting problems for different entities (solar, wind, prices, etc.).
- We future-proof the input structure through flexible configuration scripts to easily incorporate newer and powerful forecasting algorithms.
- We customize the output of the scoring process to include percentile bands and explainability flags while providing information on input data drift or outlier to patterns observed in the domain.

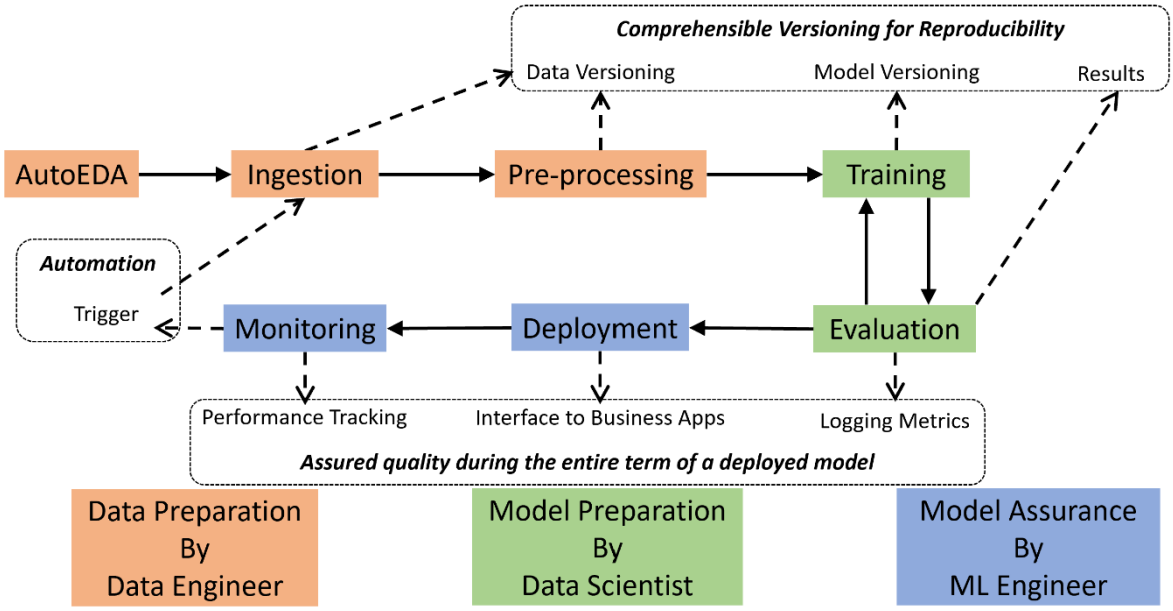

Figure 1: Forecasting Framework Lifecycle and Value Proposition

2. Version release details

Below we detail the location of the forecasting framework assets –

- Forecasting Framework: Release 1.0
- Location: Forecasting Framework DevOps
- Release Date: July 8, 2022
- Repository type: Private
- Release number: 20220712_1.0
- Release Notes: DevOps Location
- Documentation: Link

In the current release, the EDA scripts are kept in a separate repository as they are not integrated in AML. You can find the temporary EDA scripts in another DevOps repository.

3. Azure Components

Figure 2 describes the architecture of the forecasting framework. The overall setup runs on Azure Machine Learning (AML) and the following linked services. AML allows for data and model versioning, logging, etc. Moreover, we build pipelines that get executed on containers with specific environments suited to run the diversity of time series forecasting algorithms we support.

- Blob Storage – To store input data and processed data.
- Azure Container Registry – To hold the different models post the training process.
- Key Vault – To maintain secrets.

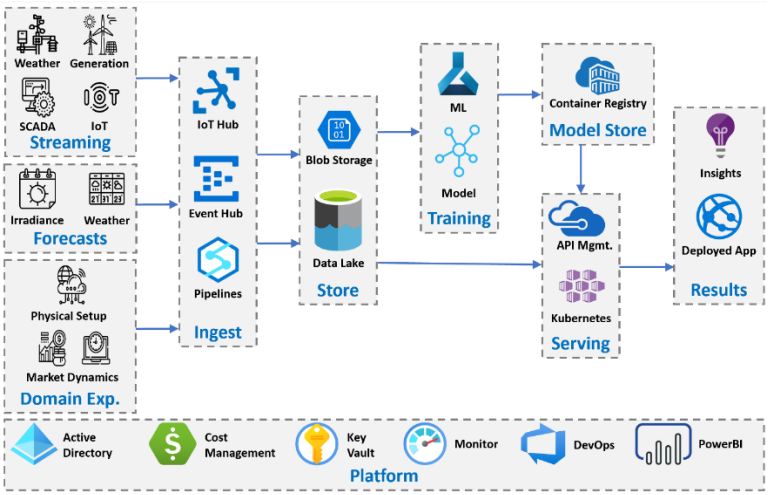

Figure 2: Forecasting Framework Architecture illustrated through the Azure Components used

In this release, we worked on enabling the following features on the AML pipelines -

- **Heterogenous compute clusters:** Different state-of-the-art algorithms have varying computing needs. However, AML only allows the creation of homogenous compute clusters. Thus, while executing a training pipeline, we partition algorithms into various buckets based on their computing needs and allocate clusters with appropriate compute nodes for better utilization of resources to get results faster.
- **Reproducibility and performance diagnostics:** Along with the data and model versioning capability provided by AMl, we also log the configurations for complete reproducibility of the experiment runs. Moreover, we add logging to diagnose errors and track model performance.
- **Domain-specific feature engineering:** We provide a suite of feature engineering methods catering to the needs of domain data scientists. For example, we offer trigonometric transformations to handle vector quantities such as wind speed.
- **Domain-specific interpolation:** We provide a wide variety of customizable interpolation techniques that are important to capture the diversity of variables in P&U forecasting scenarios.
- **Algorithm-specific environment setup:** We codify the dependencies for various algorithms and create customized environments for easy deployment of containers that perform data preprocessing and model training.
- **Configurable training setup:** Our configuration file allows ML engineers to tweak training setup (hyperparameters, loss function, etc.) without any code modifications.

We intend to deploy models on the following Azure services – i) Azure Kubernetes Service, ii) Azure Functions. The AutoEDA currently needs to be run on a VM. However, we will be integrating this part also in AML. Ultimately, AutoEDA will be a precursor to the preprocessing step in our MLOps.

4. Usage

We enable the setup through a standard configuration file specified in a YAML format. The YAML file simplifies the process of specifying the forecasting task without compromising on the flexibility offered by custom scripting (iPython notebooks). Moreover, the structure of the YAML file allows the configuration of specific components relevant to domain data scientists. These include customizable feature engineering, data interpolation, and training-specific setups such as hyperparameters ranges and loss functions to optimize during training. In particular, the YAML configuration file contains the following top-level keys

- **SourceConfig:** This field contains information on the location of the dataset. Specifically, one can categorize data into the following three categories – entity data, past covariate data and future covariate data. **Figure 3** provides a sample values that are specified in this field.

```
#Copyright (c) Microsoft. All rights reserved.
SourceConfig:

  source_dict : {"Owner":"XYZ", "Site name":"Site1", "Resource Type":"Solar" ,
                 "Resource Configuration":"Fixed Axis","Manufacturer Name":"Man1",
                 "Model Name" : "Model1",
                 "Latitude":44.1381958, "Longitude":87.3146306}

  scripts_directory : './preprocess'
  container_name : "blobdata"
  secret_name : "forecastblobkey"
  account_name : 'forecastdatapoc'
  datastore_name :  'hornsdale_datastore' # "ds_forecast"

  # entity details like price , power etc...
  entity_data : 'input/entity/entity_data.csv'
  # Hisotrical entity details like windspeed , wind direction , IOT data etc.
  past_covariates : 'input/past_covariates/past_covariates.csv'
  # Forecast API data like windspeed , wind direction , solar irradiance etc.
  future_covariates : [{'solcast_forecast':'input/future_covariates/future_covariates.csv'}]

  datetime_col_applicable : 'applicable_date' # will be there in all three files.
  datetime_col_available : 'available_date'  # will be there in future covariates
  source_datetime_col : 'DateTime'
```

Figure 3: SourceConfig sub keys

- **PreprocessConfig:** Data cleaning and preparation is a crucial step before model training. In this step, we can perform feature engineering, dataset alignment, imputation for handling missing values, etc. **Figure 4** provides a sample values that are specified in this field.

Figure 4: PreprocessConfig sub keys

- **PipelineConfig:** As discussed earlier, we use AML Pipelines to execute model training in parallel. Specific details on pipeline includes environment and python endpoints for various containers running individual ML algorithms. **Figure 5** provides a sample values that are specified in this field.

Figure 5: PipelineConfig sub keys

- **TrainConfig:** Forecasting specific details along with hyperparameters for various algorithms are specified in the train configuration field. **Figure 6** provides a sample values that are specified in this field.

Figure 6: TrainConfig sub keys

- **ScoringConfig:** In the scoring configuration, the user selects a trained model along with the specific setup details. **Figure 6** provides a sample values that are specified in this field.

```
ScoringConfig:
  modelName : "DartsTFT"
  sourceFileName : "darts_scoring_data_future.csv" #"scoring_test_data.csv"
  requiredColumns : ["hist_WindSpeed10m"]
  targetColumn : "Power"
  futureCovariates : ["solcast_forecast_windSpeed"]
  dateField : "DateTime"
  modelPath : "outputs/"
  # Parameters to get predictions
  parameters : {
      "DartsTFT" : {
        "n" : 6
        ,"forecast_horizon" : 24
        ,"verbose" : False
        ,"last_points_only" : False },
      "DeepMC" : {
        "horizon" : 6
        ,"trunc" : 512
        ,"lookback" : 24
        ,"freq" : "60min"
        ,"frequency_in_minutes" : 60
        ,"frequency_duration" : "min"
      }
    }
```

Figure 7: ScoringConfig sub keys