Winds of Change: A Machine Learning Approach to Predicting Wind Power Generation

Jannik Elsäßer[1]

[1]**jels@itu.dk, Student No. 20355**

ABSTRACT

This paper explores the use of machine learning models to forecast wind power production using weather data, in the context of managing the variability of renewable energy sources. Specifically, it examines data from Orkney's renewable power generation and UK MetOffice weather forecasts to predict wind power production. The study highlights the advantages and limitations of machine learning models for this task, and presents wind speed transformation techniques such as empirical wavelet transformation to gain more information from the data. Two grid searches with their respective pipelines are used to predict actively or non-actively managed wind turbine power generation, and the paper provides a detailed description of pre-processing and transformation techniques employed. The research concludes that machine learning models are capable of accurately predicting non-actively managed wind turbine power generation. This work is part of my submission for Assignment 1 for the Large Scale Data Analysis course in the BSc. in Data Science 4th semester program at the IT University of Copenhagen.

Keywords: Feature engineering, Wind energy, Time-series analysis, Data pre-processing, Energy production forecasting, Machine learning algorithms

INTRODUCTION

The growing demand for sustainable energy has led to the use of renewable sources like solar and wind power, but their variability poses a challenge for balancing energy supply and demand. Accurately forecasting power generation using weather data is important for efficient grid management and reducing reliance on traditional power sources. Machine learning models have proven effective in this task, and this research paper explores their advantages and limitations.

Using data on Orkney's renewable power generation sourced from Scottish and Southern Electricity Networks SSEN, and weather forecast data provided by the UK MetOffice MetOffice, this research paper aims to forecast wind power production on the Orkney Islands, an archipelago with a population of approximately 20,000 people. Orkney is a world leader in wind energy, producing around 120% of its annual energy consumption from wind turbines. Accurately predicting wind power production is crucial for effective energy management and could help Orkney to become a model for sustainable energy production.

UNDERSTANDING THE DATA AND PIPELINE DESIGN

The assignment objective was to predict the total power generation based on two features, wind speed and wind direction. In that sense, it was recommended to disregard the ANM and Non-ANM columns which were part of the original SSEN dataset. However after understanding the meaning of ANM, Active Network Management, I realized that it would be a fundamental flaw, to ignore these columns, because they are what the total energy generation is made up of. Simply put, the total power generation is the sum of the actively and not actively network managed power generation sources. Active network management power sources, are sources which are actively managed (i.e. turned on/off or efficiency improved/worsened) in accordance to the current energy demand in the network.

Therefore, it is incredibly difficult to predict the ANM wind turbine power generation, since we do not have any information on energy demand/consumption in the network.

However, it is still possible to predict Non-ANM wind turbine energy production, and in fact very accurately. Moreover, to overcome the limitations of ANM wind turbine power generation, I have added

the time of day and date as a feature, to help see patterns in power consumption.

It is for these reasons, that my ML pipeline has two gridsearches, each with their own pipeline, which each predict ANM or Non-ANM, and are then funneled together into a sum, to predict the total power generation (see Figure 1.).

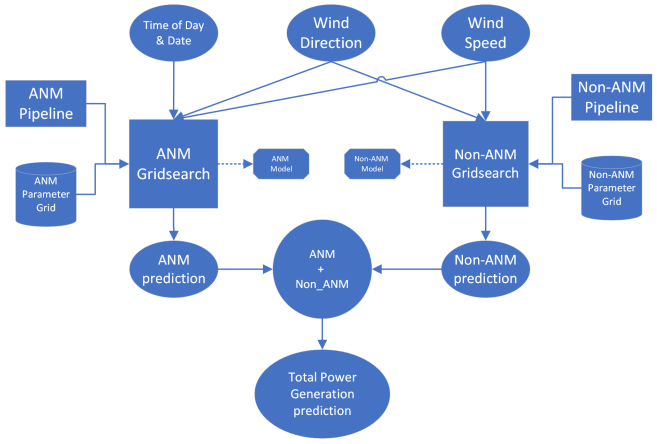

Figure 1. A diagram of the entire model

PRE-PROCESSING

Before porting the data into the different machine learning models, the data was transformed.

Wind Speed Transformation

Wind speed is by far the most important feature when predicting the power generation. To help the models gain as much information as possible from it, the wind speed was transformed the using a several different methods.

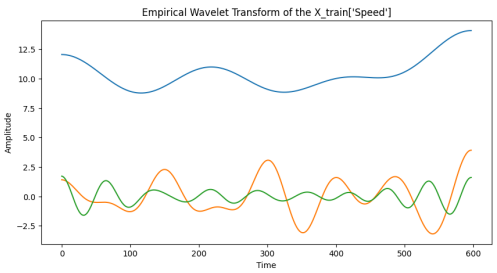

Figure 2. Empirical Wavelet Transformation of the X_train['Speed'] feature, each line is a different component

Method 1. was using the `StandardScaler()`. The sklearn `StandardScaler()` is a transformer that scales and centers the data such that it has zero mean and unit variance. This is useful when using machine learning models that are sensitive to the scale of the input features, for example linear regression, logistic regression, ridge regression and lasso regression (all of which were used at some point).

Method 2. was using the sklearn `PolynomialFeatures()` transformer. The transformer is used to generate polynomial features from the original features of a data set by creating interaction terms, which can be useful for capturing non-linear relationships between the features and the target variable in a linear regression model. There seems to be a non-linear relationship between the power generation and wind speed. In some time frames this seems power-curve or sigmoid like, and in others like below it seems the relationship is described more in a inverted quadratic function. Whatever the case, the `PolynomialFeatures()` allows for this.

Method 3. was more unorthodox, and based on findings provided in the research project present by students at DTU, Peleguer and in the research paper Liu et al.. The approach was to run an empirical wavelettransformation. Empirical wavelet transformation (EWT) is a signal processing method that decomposes a time-series signal into a set of intrinsic oscillatory modes or components. EWT extracts the most oscillatory patterns in the signal using a band-pass filter bank with a variable center frequency and bandwidth. EWT-derived features can be used to classify and analyze signals in various applications,

making it a valuable tool for feature extraction in machine learning. Therefore I coded a custom sklearn transformer using the `ewtpy` python library Carvalho:

```python
class EmpiricalWaveletTransform(BaseEstimator, TransformerMixin):
    def __init__(self, level=5, log=False):
        self.level = level
        self.log = log

    def fit(self, X, y=None):
        return self

    def transform(self, X, y=None):
        X = np.asarray(X)
        elems = X.size
        X = X.reshape(elems)
        ewt = ewtpy.EWT1D(X, N=self.level, log=self.log)
        components = []
        for i in range(len(ewt)):
            components.append(ewt[0][:, i].ravel())
        return np.asarray(components).T
```

Listing 1. EWT Sklearn Transformer

By default the level, or maximum amount of components is set to 5. This was decided based on a grid search (more on this in the machine learning section).

The final method which I used to extract as much information as possible, was the `WindToComplexTransformer()` (see fx.py, transformers section). This was also a custom transformer, which takes the wind speed and direction, and converts it into a complex number. Also known as a phasor, it captures both magnitude and direction information in a single variable, enabling machine learning models to better capture the relationship between wind conditions and their effects on energy production. This technique has been widely used in the field of renewable energy forecasting to improve the accuracy of wind energy prediction models .

Wind Direction Transformation

The wind direction feature was also transformed. It was provided was in a discrete format, i.e. in compass directions. It is debatable if the wind is a discrete feature, when it is only provided in such a format. However I would make the argument that this is a fundamentally wrong assumption, since wind itself is a continuous feature. Just because it is not given to us as such, does not mean we should interpret it as such. Out of interest however, I did include a one-hot encoder feature transformation in the pipeline, sklearns `OneHotEncoder()`, to see if this interpretation really did have any effect on prediction accuracy. Nonetheless, it was never chosen by the gridsearch for either ANM and non-ANM ML models.

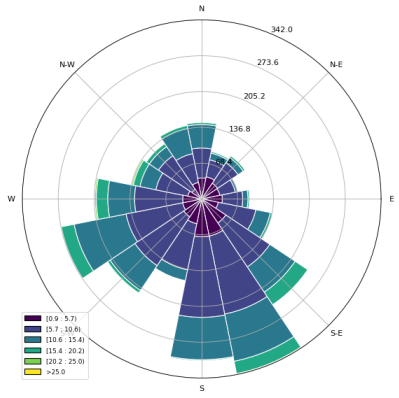

Figure 3. Wind rose of directions and speed of wind for one year (data pulled 2023/02/10)

The other two transformers I included where `WindDirectionMapper()` (see fx.py) which maps the compass direction strings to compass degrees, and `CompassToCartesianTransformer()`.

Mathematically it is difficult to deal with a compass degree that is changing from 360 to 0 degrees. This can be over come by coordinate transformation to cartesian coordinates, .i.e the `CompassToCartesianTransformer()`, or converting to complex numbers, using `WindToComplexTransformer()`. In Cartesian coordinates, the directional information is represented by two orthogonal components, such as x and y coordinates. The x and y coordinates can be standardized to have zero mean and unit variance, making them suitable for use in various machine learning algorithms.

The Time Transformer

The timestamp, or index column, was also transformed using the custom `TimestampTransformer()`. It was created to aid the prediction of the active network managed power generation, with the general

idea being that the ML models would see patterns based on the time and date, for example the energy demand sharply rising and slowly falling in the mornings (high domestic power consumption), and a higher average energy demand on week days than on weekends (lower industrial power consumption on weekends).

Aligning both Data-sets

Aligning both the weather forecast data, and the power generation data was a not so simple task, since the weather data provided was in 3 hour observations/forecasts, and the power generation was tracked in minutes. Therefore, the power generation data was re-sampled in 3 hour means. One could have also simply taken the tracked power generation from every third hour, however this could have resulted in a higher degree of fidelity in the power generation data, than in the weather data. In a best case scenario one would use a weather data set which has observations/forecasts in a similar temporal variability, to the power generation data.

MACHINE LEARNING

After re-sampling the power generation data in 3 hour slots, and combining it with the the forecast data, making sure that all data points are connected to a timestamp, it is then fed into two sklearn (one for ANM and one for non-ANM) `GridSearchCV` objects, which use a machine learning pipeline as a estimator, a sklearn `TimeSeriesSplit()` cross validation splitter, and a parameter grid:

```
anm_pipeline = Pipeline(steps=[
    ("col_transformer", ColumnTransformer(transformers=[
        ("time", None, []),
        ("Speed", None, ["Speed"]),
        ("Direction", None, ["Direction"]),
        ], remainder="drop")),
    ("model", None)
])
anm_params = {
    'col_transformer__time' : ["drop", None, fx.TimestampTransformer()],
    'col_transformer__Speed': [None, StandardScaler(), PolynomialFeatures(), fx.
     EmpiricalWaveletTransform(level=5)],
    'col_transformer__Direction': ["drop", fx.WindDirectionMapper(), fx.
     CompassToCartesianTransformer()],
    'model': [
        LinearRegression(),
        MLPRegressor(hidden_layer_sizes=(150, 150), activation='tanh', solver='sgd'),
        SVR(kernel='rbf', gamma='scale', C=1.0, epsilon=0.1),
        HuberRegressor(epsilon=1.35, alpha=0.0001),
        RANSACRegressor(min_samples=0.1, max_trials=100),
        GaussianProcessRegressor(alpha=0.1, kernel=RBF())
    ]
}
pipeline = Pipeline(steps=[
    ("col_transformer", ColumnTransformer(transformers=[
        ("Speed", None, ["Speed"]),
        ("Direction", None, ["Direction"]),
        ], remainder="drop")),
    ("model", None)
])
tscv = TimeSeriesSplit(n_splits=5)
anm_gridsearch = GridSearchCV(anm_pipeline, anm_params, cv=tscv, scoring='neg_mean_squared_error',
      n_jobs=-1, verbose=1)
```

Listing 2. ANM Model Pipeline

This pipeline construction has the significant advantage, that I can then use a `param_grid` python dictionary inputing different column transformers and models, and then let grid search find the best ones for each model, every time it is run.

Notice that one of the options for the wind direction column transformer and the time transformer, is ”drop”. This means that grid search can also automatically decide, if it is beneficial to have direction or time as a feature.

As described before (see Figure 1.), the same process is run on the non-ANM pipeline (it differing to the ANM pipeline only on the fact that it does not include time of day and date as a feature, ever), and both girdsearch best estimators are saved. They are then retrained on the entire dataset, and used to predict on the forecasted weather. See below some of the data plotted with the model, the actual power generation, and the forecast.

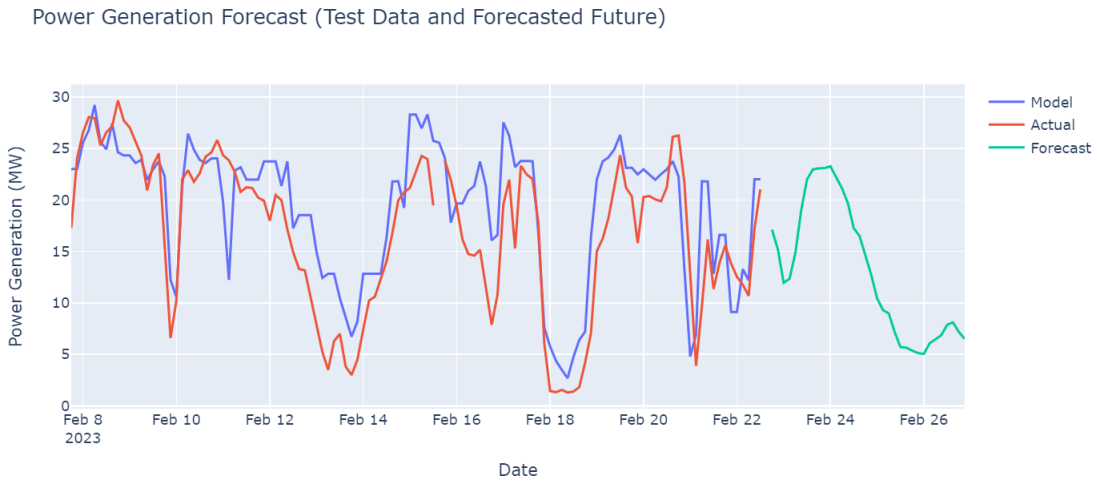

Figure 4. Test Data, with Actual, Model and Forecasted Power Generation [MW]

RESULTS

In general, both grid searches return the MLP regressor as the best estimator, in combination with a standard scaler transformation on the wind speed feature, and the compass to cartesian custom transformers. The MSE score is very dependent on the data, in a best case scenario the model scores an overall MSE score of 23, and averages usually around 28. In tests it consistently performs better than a ML pipeline which does not predict ANM and non-ANM, but total directly (using the same hyper-/parameters). Visit here to see which model it is currently selecting, it's MSE score and power generation predictions for the next 5 days.

DISCUSSION

Unfortunately, the custom empirical wavelet transformer does not seem to improve model performance, nor does the wind to complex transformer. The issue with the wind to complex transformer seems to be caused by a faulty implementation. For EWT to work correctly one must always read from the same time of day, which is quite difficult since the data is constantly updating. Liu et al. uses a combination of a long short term memory neural network and an Elman neural network, where each deals with different types of frequency components. They are therefore able to more effectively use it. However in future work it would certainly be worthwhile to implement these.

It would also quite possibly be worthwhile to do an analysis on a larger historical data-set, and see if trends are found annually/seasonally, and then include this as an additional feature. This is something that EWT could be used for, since with it one can find the component which corresponds to the long term trends.

REFERENCES

SSEN. Active network management. URL https://www.ssen.co.uk/link/d4476c9a50a24e19b2b13ad44dd41c68.aspx.

MetOffice. Westray airfield (orkney islands) weather. URL https://www.metoffice.gov.uk/weather/forecast/gftcsumwq.

Marcos Ivorra Peleguer. windpowerforecastDTU. URL https://github.com/Marcoscos/windpowerforecastDTU. original-date: 2022-01-03T16:35:03Z.

Hui Liu, Xi-wei Mi, and Yan-fei Li. Wind speed forecasting method based on deep learning strategy using empirical wavelet transform, long short term memory neural network and elman neural network. 156:498–514. ISSN 0196-8904. doi: 10.1016/j.enconman.2017.11.053. URL https://www.sciencedirect.com/science/article/pii/S0196890417311056.

Vinícius Carvalho. ewtpy - empirical wavelet transform in python. URL https://github.com/vrcarva/ewtpy/blob/f9ff38bc5cd7859eea58c5b29099f8a503792b63/ewtpy/ewtpy.py. original-date: 2019-03-27T22:46:24Z.