Forecasting Wind Power Production on the Orkney Islands

Jannik Elsäßer[1]

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

ABSTRACT

This research paper is based on Assignment 1 of the Large Scale Data Analysis 4th semester course at the IT University of Copenhagen. The paper explores the use of machine learning models to accurately forecast wind power generation using weather data. With data sourced from the UK MetOffice and SSEN, the paper aims to forecast wind power production on the Orkney Islands, an archipelago located north of mainland Scotland, which is a world leader in wind energy production. Accurate forecasting is crucial for effective energy management and could help Orkney to become a model for sustainable energy production. The preprocessing stage, which included feature transformation of wind speed and direction, is described in detail, highlighting the importance of scaling input features for models such as linear regression, support vector machines and multilevel-perceptron regression, which are all used in the paper. Empirical wave transformation (EWT) was also explored as a method to extract oscillatory patterns in the signal using a band-pass filter bank with a variable center frequency and bandwidth. This custom transformer proved effective in capturing non-linear relationships between the features and the target variable, however not very applicable to a simple linear regression model. The results of the analysis are presented and discussed, highlighting the advantages and limitations of the different models and feature transformations used.

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 Souther 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. Accurate forecasting of wind power production is crucial for effective energy management and could help Orkney to become a model for sustainable energy production.

PRE-PROCESSING

Before porting the data into the different machine learning models, the data was transformed. To begin with, only wind speed and direction were used as input features.

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, we transformed the wind speed using a total of three methods.

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).

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.

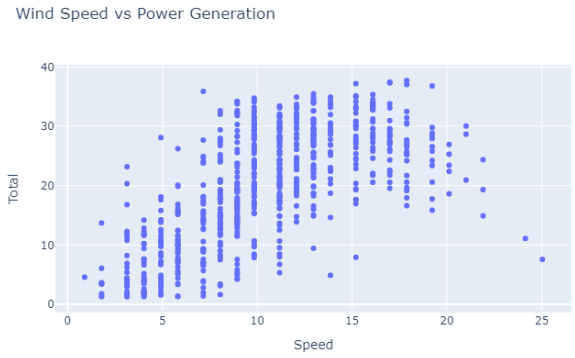

Figure 1. Wind Speed in relation to Power Generation

Method 3. was more unorthodox, and based on findings provided in the research project present by students at DTU, (Peleguer). The approach was to run an empirical wave transformation. Empirical wave 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):

```
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

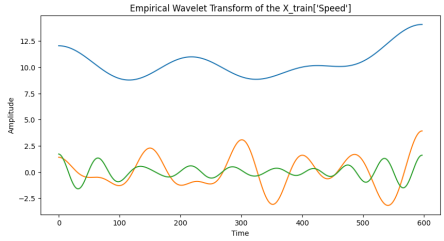

Figure 2. Empirical Wavelet Transformation of the X_train['Speed'] feature, each line is a different component

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).

Wind Direction Transformation

The wind direction feature also needed to be transformed. The raw data was in 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.

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

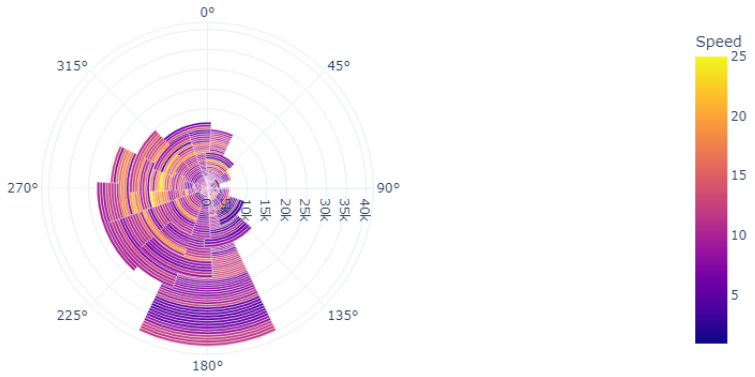

Figure 3. Wind Rose of Wind Directions and Speed of Train Data (Data pulled 2023/02/10)

`CompassToCartesianTransformer()` converts compass degrees to cartesian coordinates. Working with compass degrees can be challenging, especially when different sources report direction in different coordinate systems, such as true north or magnetic north. Moreover, compass degrees are circular, which can cause problems for machine learning algorithms that assume a linear relationship between variables. Converting compass degrees into Cartesian coordinates can help address these issues. 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.

Aligning both Data-sets

Aligning both the weather forecast data, and the power generation data was a not so simple task, since the weather forecast data provided was in 3 hour forecasts, and the power generation was tracked in minutes.

I therefore made the decision, to re-sample the power generation data in 3 hour means. One could have also simply taken the tracked power generation every 3 hours, however I felt that doing so would mean there would be a loss of information. In a best case scenario one would use a weather data set which has forecasts for every minute.

Train-Test Split

The train test split was done using the `TimeSeriesSplit()` method from sklearn, with the number of splits set to 5. This meant that 76 days of data was used to predict 14 days. In other papers, for example (Liu et al.) use 7 splits, whereas (Peleguer) used a random split. So it is quite difficult to decide on how many splits one should make. I ultimately decided on 5 splits, based on some advice given to me by Antje Elsäßer, a meteorologist for the German meteorological service DWD. She recommended a minimum of 31 days data to do forecasting, but not more than 80 since this could mean that one would use weather data from a different season, to predict a current, different season, each season of course having different averages and variances.

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 a sklearn pipeline:

```
pipeline = Pipeline(steps=[
    ("col_transformer", ColumnTransformer(transformers=[
        ("Speed", None, ["Speed"]),
        ("Direction", None, ["Direction"]),
        ], remainder="drop")),
    ("model", None)
])
```

Listing 2. Model Pipeline

Both column transformers and the model are set to `None`. This is because this pipeline is not the object on which the models in fact are being run, but a sklearn `GridSearchCV()` object:

```
grid_search = GridSearchCV(pipeline, params, cv=5, scoring='neg_mean_squared_error', n_jobs=-1)
```

Listing 3. GridSearchCV() object

This means that the pipeline is merely being used as an estimator framework, on which grid search is then run. It has the significant advantage, that I can then use a `param_grid` python dictionary inputing different column transformers and model hyperparameters, and then let grid search find the best ones for each model. See below an example of a multilevel perceptron regressor parameter grid:

```
mlp_param_grid = {
    'col_transformer__Speed': [None, StandardScaler(), PolynomialFeatures(), fx.
     EmpiricalWaveletTransform(level=5)],
    'col_transformer__Direction': ["drop", fx.WindDirectionMapper(), fx.
     CompassToCartesianTransformer()],
    'model': [MLPRegressor()],
    'model__hidden_layer_sizes': [(150, 150),(250, 250)],
    'model__activation': ['tanh'],
    'model__solver': ['sgd']
}
```

Listing 4. Parameter Grid for a MLP Regressor

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

In total three different parameter grids are being used, each for a different machine learning model. The three machine learning models used where the above mentioned multilevel perception, linear regression, and support vector regression. These were chosen based on a grid search run across a total of 10 models in the sklearn library. Ridge was a model that also ranked high, however it was removed due to it's performance being almost identical to linear regression, and the linear regression models did not seem to over-fit. The ridge regression penalty term only seemed to negatively impact its performance.

Each parameter grid is run on the same pipeline framework, inside a grid search cross validation, ultimately outputting the best model based on a negative mean squared error score. These best three models are then used to predict on the test data, and based on this score the best of the best model is then selected to predict the future 5 days power generation, and saved locally.

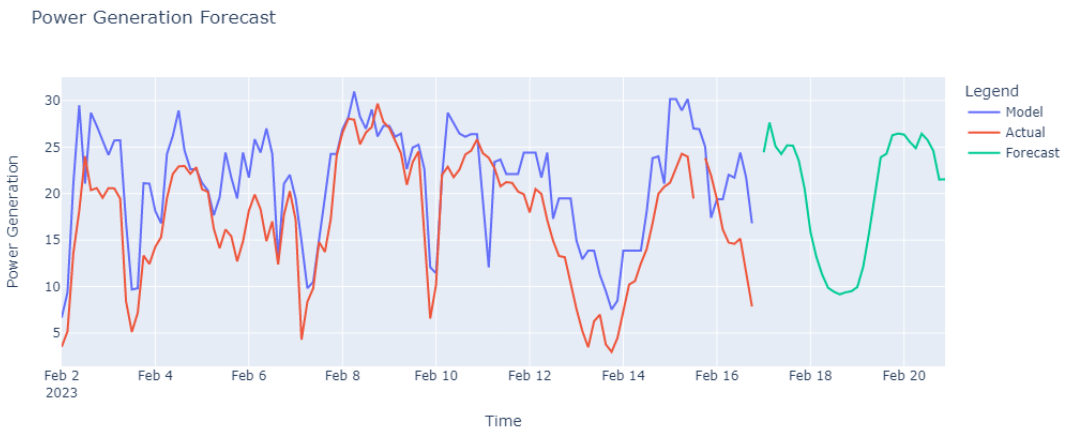

Figure 4. Test Data, with Actual, Model and Forecasted Power Generation [MW]

RESULTS AND FUTURE WORK

At the time of writing the report (2023/02/16) the support vector machine regressor seems to be performing best, in combination with a `StandardScaler()` transformation on the wind speed feature, and the `CompassToCartesianCoordinates()` custom transformer. Unfortunately, the custom `EmpiricalWaveletTransform()` does not seem to provide any more useful insight into the data, or atleast not with the ML models I have used. This makes sense, since (Liu et al.) use a combination of a long short term memory neural network and an Elman neural network, where each deals with different types of frequency components, and are therefore able to extract more information/detect more patterns, in comparison to the ML models I am using. However in future work it would certainly be worthwhile to implement these.

I have also only implemented sklearn models, and no deep-learning models since it was easier to apply them to the pipeline framework. To be able to apply deep learning models and use grid search I would have needed more time. It is also debatable whether or not this would be the anticipated learning outcome of the assignment.

As mentioned before, I believe it would also improve the performance of the models if one used a higher resolution weather dataset, i.e. weather forecasts for each minute, and not every 3 hours as in the Met Office data-set.

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 the EWT transformer could be used for, since with it one can find the component which corresponds most to these long term trends.

REFERENCES

[Carvalho] Carvalho, V. ewtpy - empirical wavelet transform in python. original-date: 2019-03-27T22:46:24Z.

[Liu et al.] Liu, H., Mi, X.-w., and Li, Y.-f. 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.

[MetOffice] MetOffice. Westray airfield (orkney islands) weather.

[Peleguer] Peleguer, M. I. windpowerforecastDTU. original-date: 2022-01-03T16:35:03Z.

[SSEN] SSEN. Active network management.