# Machine Learning Applications for Offshore Wind Farms

Austin Hancock
*MSDS 7335-401*

## Introduction

Trends in net electricity generation from renewable energy have pushed these types of power generation into the forefront of energy research. With a near doubling in billion kilowatt-hours per year projected within the next 10 years, this sector is ripe for analysis. One area of interest within this sector is wind farms. These utility-scale farms began popping up in the U.S. in the early 1980's and have been met with mixed reviews. While some people champion the projects as a clean alternative to fossil fuels, others (typically those that live near the farms) believe them to be an eyesore that is more trouble than they are worth. An easy solution to this problem is to place these turbines out of sight where the nay-sayers can have no objections. Where is such a place? The ocean.

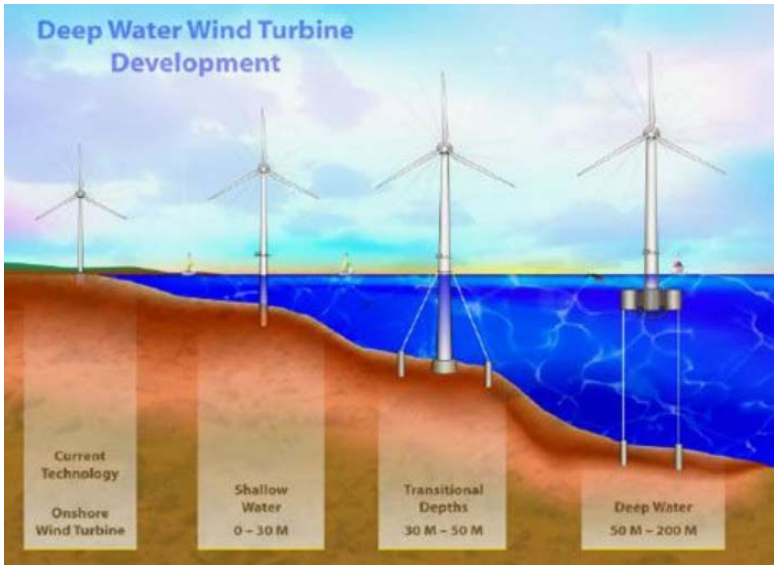


Fig 1. Turbine technology used at differing depths.

Placing wind farms offshore comes with a number of benefits that advocates and opponents of wind turbines alike can appreciate. With offshore plots being owned by the U.S. government, and not citizens, there will be no more "land grabs" or easements on properties in order to build large farms. Another benefit to offshore versus onshore wind farms is that the wind speed is easier to predict over water than it is on land. This is crucial to companies who are determining if the return on investment (ROI) of building these renewable wind farms is worth

their time and effort. An easy litmus test for a new project is that the ROI has a predictable timeframe in which the company can expect to profit from this venture. With predictable wind speeds, and therefor power output, the expense of building turbines offshore at different depths [Figure 1] can be weighed against potential revenues over a number of years.

To perform my analysis, I will pull data from multiple data sources, perform any necessary feature extraction, clean and combine the disparate sources into a single dataframe, then create multiple models [Figure 2].

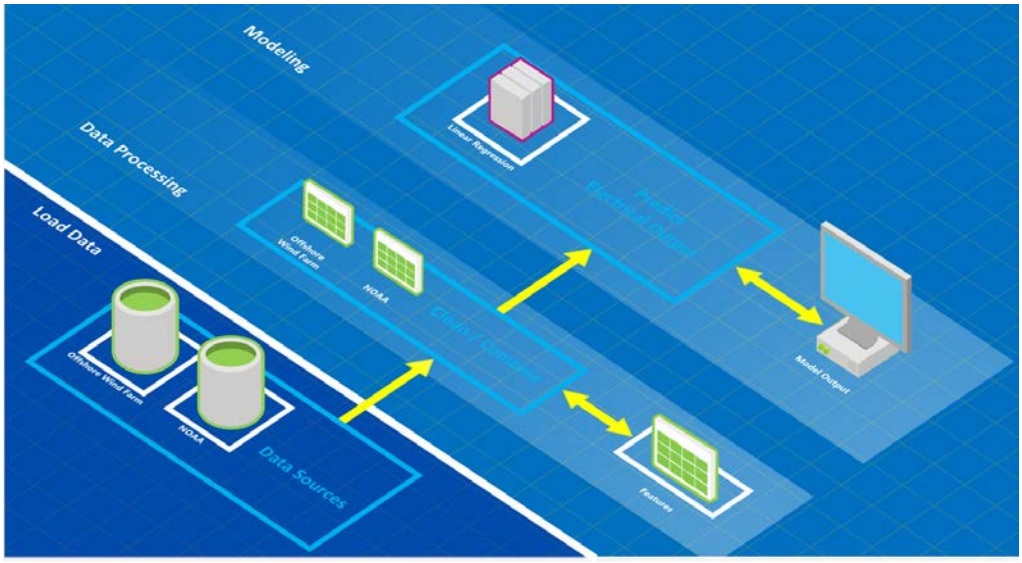


Fig 2. Project workflow.

**Data Preparation**

The main dataset for this project came from TheWindPower.net, a site that compiles wind farm data that is then used by the European Union's Maritime Affairs office. This dataset consists of 25 variables, such as latitude, longitude, ocean depth, and turbine hub height on 724 wind farms globally.

When looking at the sample data provided by The Wind Power, I knew that some data was missing but I thought that through the use of the other variables I would be able to derive these missing values. Unfortunately, it was only after I had bought the dataset and could access all of the rows that it became apparent that I could not accurately fill the missing values based on their categorical attributes. To correct for the missing values, I instead used the median value of the variable for missing data. This is far from ideal but worked as a way to get the data into a format suitable for modeling.

Through the use of a tool called Geoplaner [Figure 3] I was able to use the wind farm's latitude and longitude coordinates to fill missing depth measurements. This exercise in filling missing data revealed that the location accuracy of some of these windfarms was suspect. I had to

remove some of the farms due to their being located on land (not possible for offshore wind farms) or when their depth measurements were not possible (e.g. too high out of the water to be on a raised platform).

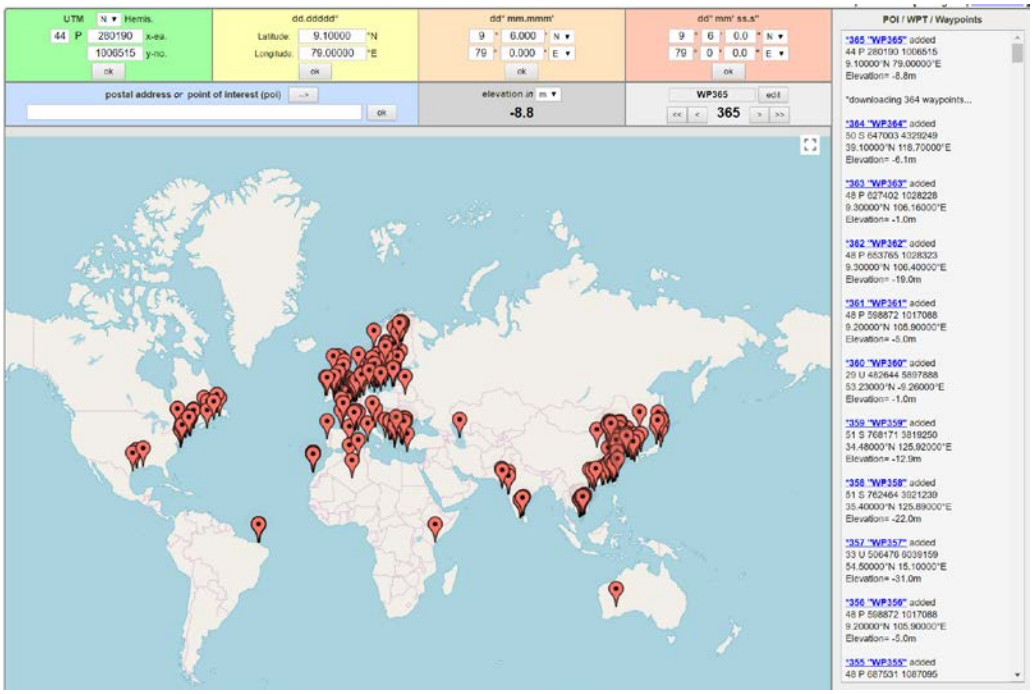


Fig 3. Lat/Lon coordinates of wind farms to find depth (m).

With the missing values taken care of, I proceeded to finish the rest of the exploratory data analysis (EDA); outlier detection and removal, data transformation, data scaling, feature extraction, and one hot encoding for categorical variables. Upon EDA completion, the dataset used for my models had changed from its original 25 variables for 724 wind farms to its final shape of 123 variables for 640 wind farms.

**Modeling**

The following is a list of the machine learning models I used and an assessment of their performance.

I. Linear Regression / KMeans Clustering

To begin my analysis, I ran a linear regression model to determine which variables would be useful in predicting wind farm power output. From this, I found that the Depth variable would be most predictive. Because there are different types of turbines that can be used at different depths [fig 1], I thought this would be a good place to use unsupervised clustering and then compare results to turbine type. In figure 4, you can see that there is good separation between depth, however, the different clusters do not distinguish themselves well in terms of differing power outputs. From the silhouette

plot, again we see good separation between the clusters, but the widths indicate (correctly) that the clusters vary greatly in number of observations.

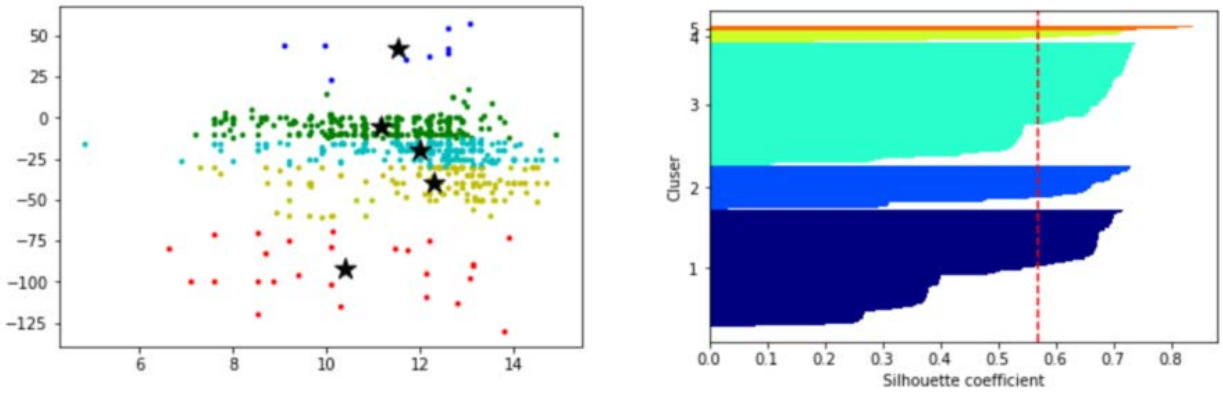


Fig 4. Scatterplot and silhouette plot of KMeans clustering of Depth (m) and Total power (kW).

Since there were no natural classifiers within the dataset (too many unique values in categorical variables), I created my own: output per turbine. With this variable, I wanted to classify wind farms on the same scale, (turbine counts range from 1 to over 200), so I divided total power output by the number of turbines. Then, using the distribution of this new variables, I created 5 classification scales to be used for supervised classification.

## II. Random Forest

Using my new variable, I trained a Random Forest Classifier. The model was created using 100 trees and a 70/30 split for training and testing. The final split counts were 448 observations for training and 192 observations for testing. The model misclassified 80 of the samples leaving us an accuracy of 58%. From the confusion matrix in figure 5, you can see that the 3$^{rd}$ class was the largest and it tended to pull other classes into it causing the greatest number of misclassifications. As we progress through the models below, we will see the influence of this class lessen, however, without a larger pool of observations from the parent dataset this will continue to be a lurking issue.

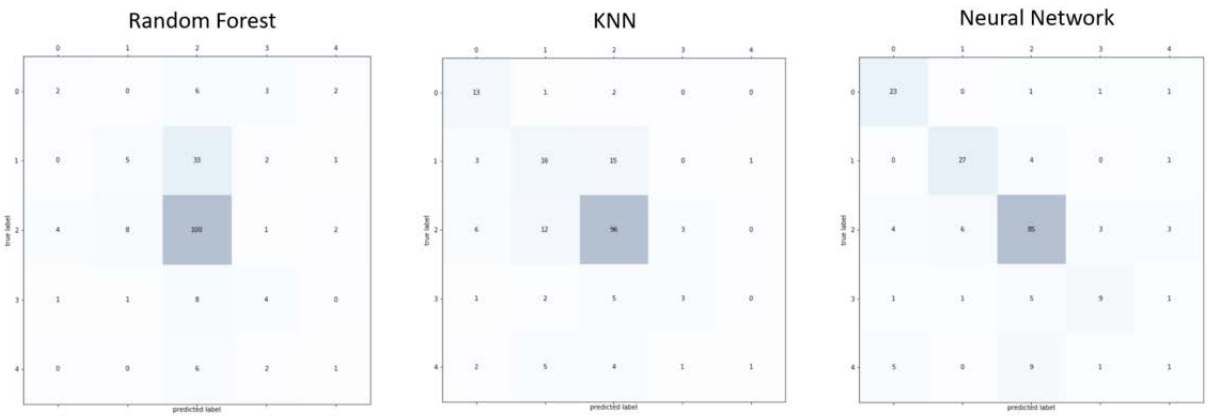


Fig 5. Confusion matrices for the random forest, k-nearest neighbor, and neural network classifiers.

III. K-Nearest Neighbors

The next model I built was k-nearest neighbors (KNN). Like the random forest model, I did a 70/30 split into 448 observations for the training set, and 192 observations for the test set. For the number of neighbors, I used 5. With a classification accuracy of 67% on 63 misclassified samples, the KNN model did considerably better than our random forest model. From the confusion matrix in figure 5, you can see that while the 3rd class still produces the most influence over classification. Even with the pull from the 3rd class, the KNN was better able to classify the smaller size groups.

IV. Neural Network

The final model used for classification of the output per turbine bins, was a neural network. The model consisted of a single-layer perceptron and 40 epochs with a train/test split of 70/30. Of the individual models, this produced the best results by only misclassifying 47 samples, giving us an accuracy of 76%. The confusion matrix in figure 5 gives a visual representation of how this model was better able to correctly classify the groups that contained a small number of observations without being heavily influenced by the larger classes.

V. Majority Vote Ensemble

Due to the small number of observations in my original dataset, the classifications for output per turbine were unequal in size and influencing the correct classification in my previous models. In an attempt to try and get any sort of high-accuracy rates, I decided to reduce my classification from multi-class to binary. Instead of 5 ranged bins, I split my response into two groups; > 33 kW/turbine and <= 33 kW/turbine. This split was down the middle of the 3rd class variable that was throwing around its weight in our confusion matrices. With this binary classifier, our counts were much more even with 343 observations above 33 kW/turbine and 297 at or below 33 kW/turbine.

Using this new variable, I created an ensemble classifier called Majority Vote. This classifier comes from the class text "Python Machine Learning" by Sebastian Raschka, and uses different classifiers to "vote" on the correct classification for an observation. The way this "voting" works is that the model uses the cross-validation scores of the different classifiers to determine which class an observation belongs too and then assigns that class to the observation that receives the most votes. To implement this ensemble method, I used a decision tree classifier, random forest classifier, and a k-nearest neighbors classifier (could not use neural network for this particular ensemble). The results can be seen in figure 6. While the accuracy scores were not as high as our individual neural network model, the benefits of the using an ensemble method can be seen in the fact that the majority vote of the three different classifiers performed better than any of the classifiers on their own.

```
Accuracy: 0.64 (+/- 0.07) [Decision Tree]
Accuracy: 0.68 (+/- 0.07) [Random Forest]
Accuracy: 0.63 (+/- 0.08) [KNN]
Accuracy: 0.69 (+/- 0.07) [Majority Voting]
```

Figure 6. Individual classifier accuracy scores and their ensemble accuracy score.

**Conclusion**

In the real world, models will not be as cut and dry as they are in the classroom. When comparing the accuracy scores achieved in this project against scores from the Iris dataset, the results can be quite discouraging. However, it did provide a great learning opportunity. The biggest issue I faced was that not all data will be helpful and clean. The dataset I used for this project was filled with holes. Some could be filled using other data sources, but others were inaccurate and, due to the proprietary nature of the subject matter, could not be corrected. In the future, it would be helpful to find better ways to fill the missing data other than using the median value, as I did for this project. Another addition that I believe to be crucial in determining the ROI for offshore wind farm investment would be wind speed by lat/lon (due to the inaccuracies within this dataset's lat/lon variables, along with time constraints, I did not bring in the wind speed data). Overall, the missing data and lack of influence of variables on the response were detrimental to accuracy scores but, with variations in models and model parameters, I was able to test the different methods we have learned and interpret the model outputs in a meaningful way. This knowledge, in addition to bringing on other observations and variables, leaves open possibilities for continued research into machine learning for offshore wind farms.