IN5410 - Assignment 2
Group 1 - ahada, krisgrav, marstorv, simonly

Machine Learning for Wind Energy Forecasting

Table of contents

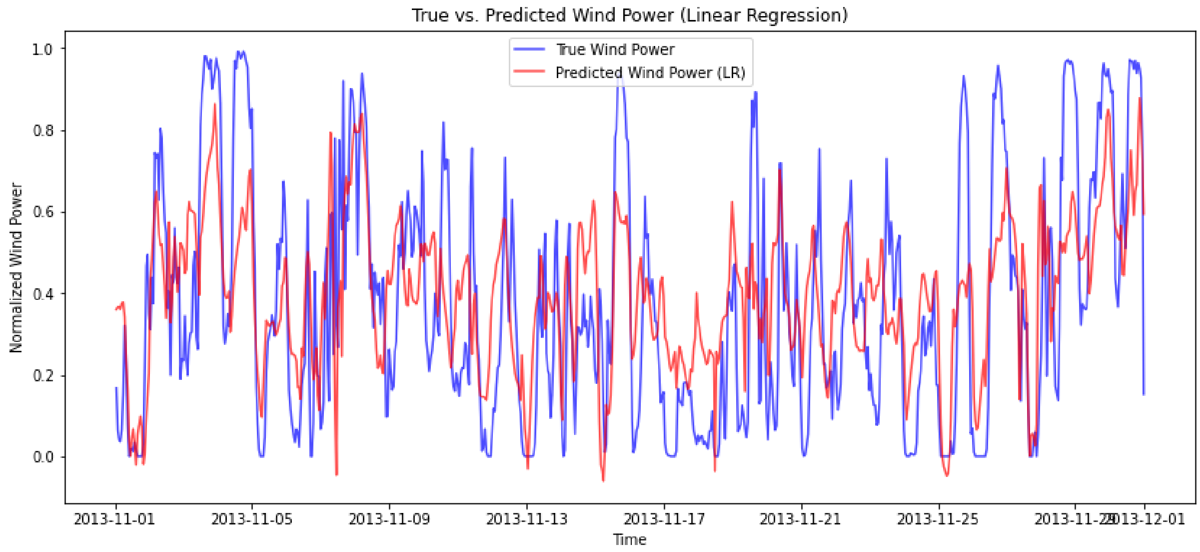
IN5410 - Assignment 2
Group 1 - ahada, krisgrav, marstorv, simonly

Task 1

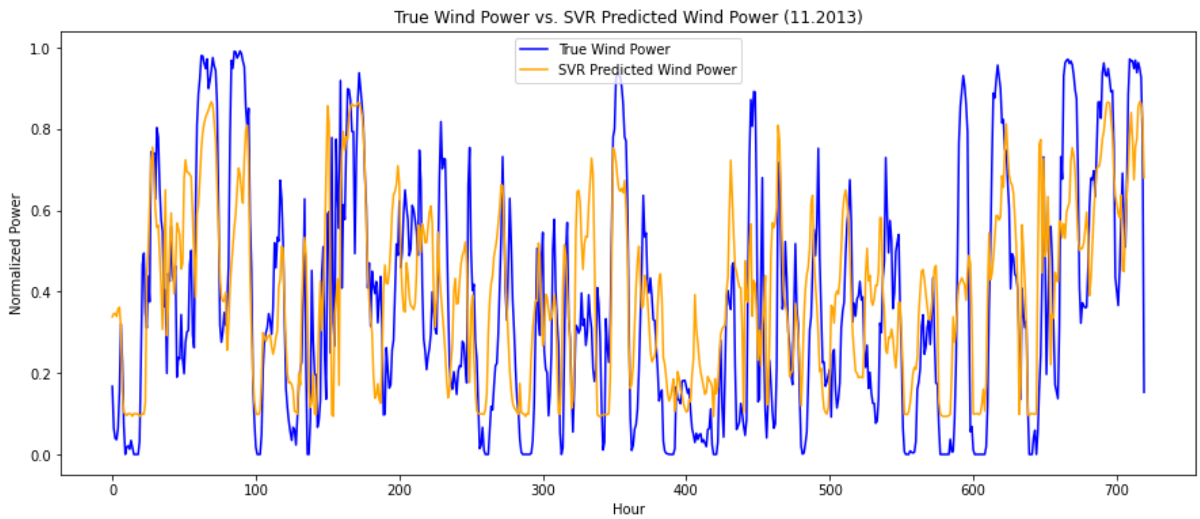
All the implementations in task 1, 2 and 3 have been written in python, using jupyter notebook. The results have been saved to files for each of the tasks with a separate results folder.

Graphs

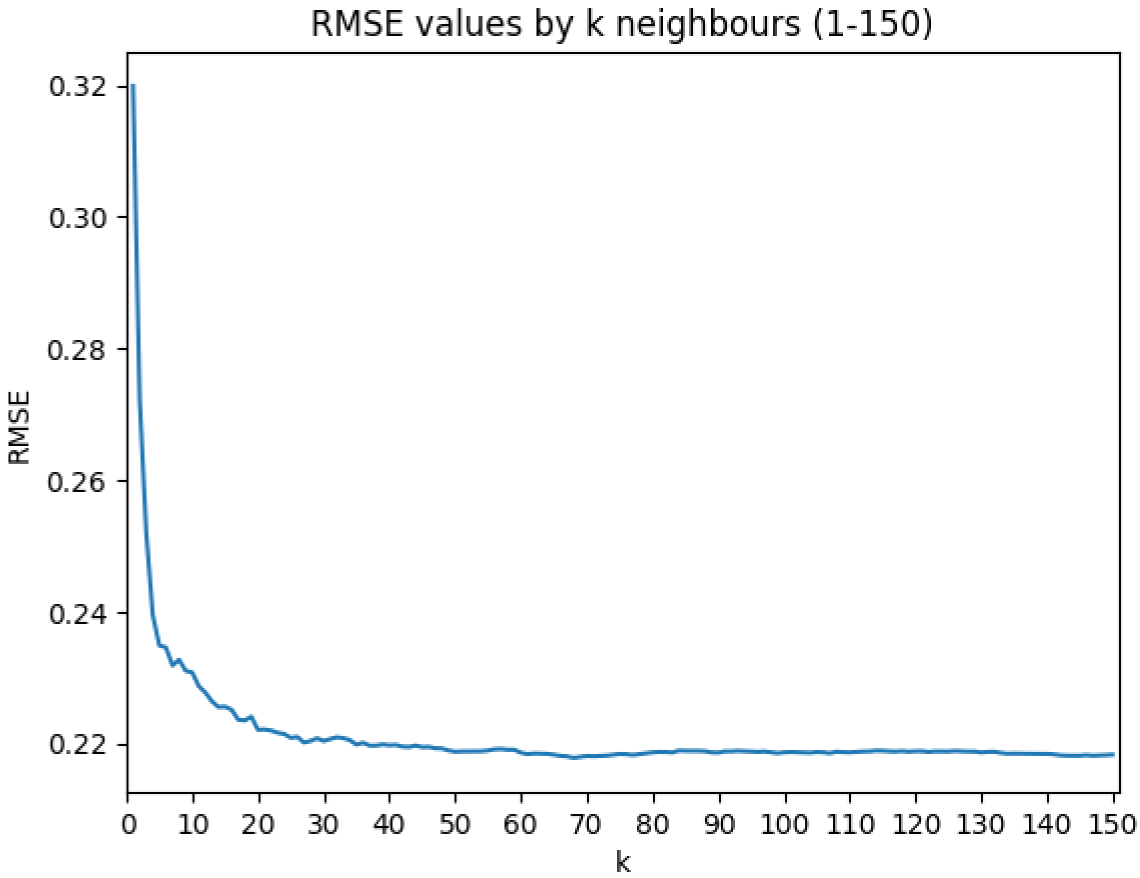
Linear Regression Graph



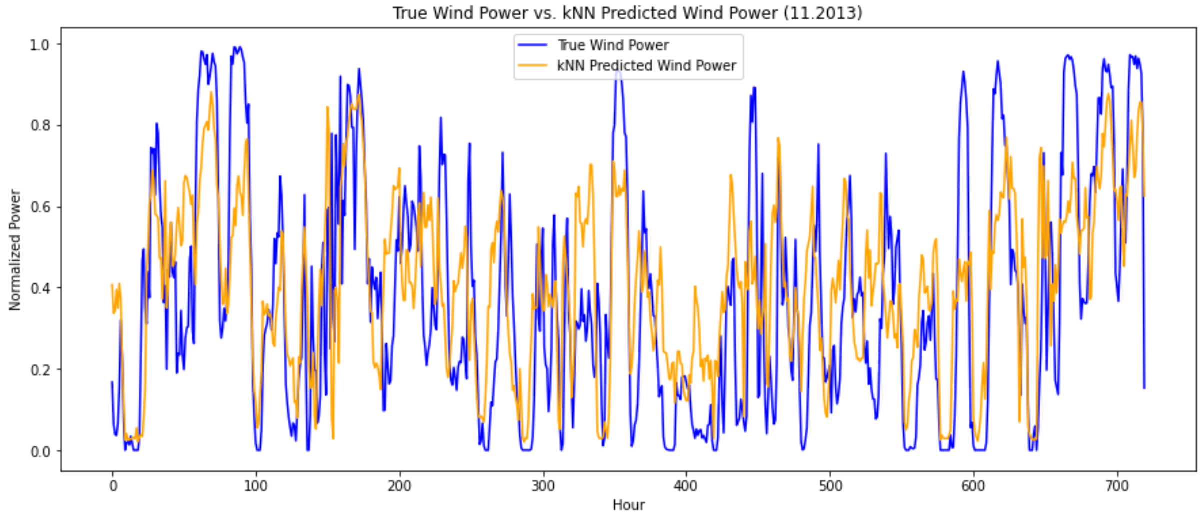
Support Vector Regression Graph



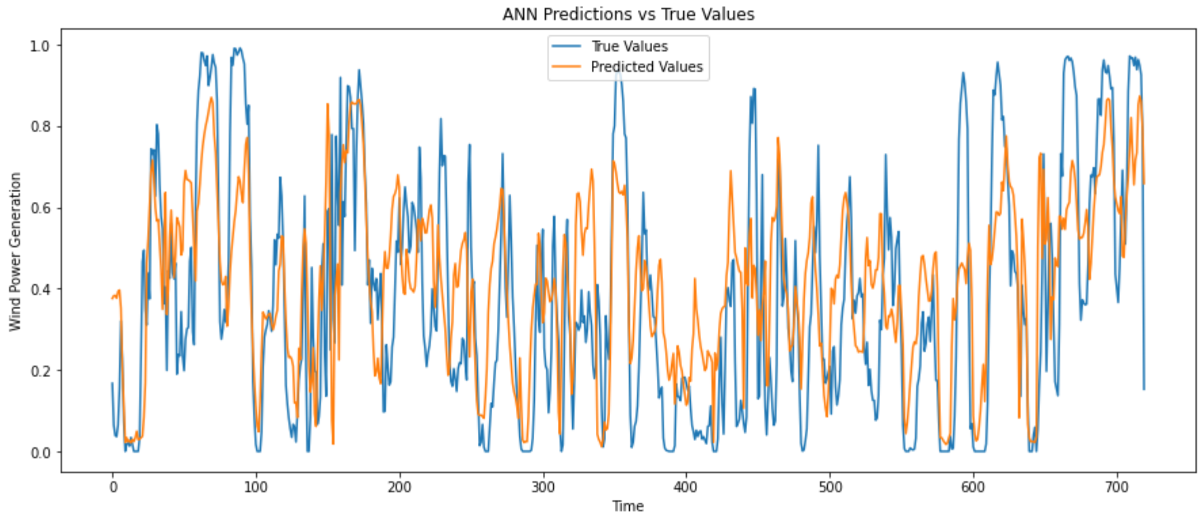
k Nearest Neighbour Graph



IN5410 - Assignment 2
Group 1 - ahada, krisgrav, marstorv, simonly



Artificial Neural Network Graph



Forecasting Accuracy (RMSE)

RMSE is a commonly used metric to measure the accuracy of a prediction model. It is a measure of the difference between predicted and actual values in a dataset.

To calculate RMSE, you use the square root of the mean of the squared differences between the predicted and actual values. This means that RMSE is a measure of the average magnitude of the errors in the predictions made by a model.

RMSE is often used in regression analysis, where the predicted values are continuous numerical values. It gives a good indication of how well a regression model is able to fit the data and make accurate predictions. Lower values of RMSE indicate that the model has a better fit and is able to make more accurate predictions.

Model	RMSE
Linear Regression (LR)	0.2164
Support Vector Regression (SVR)	0.2137
k Nearest Neighbour (kNN)	0.2186
Artificial Neural Network (ANN)	0.2175

Why the models differ

About the Models

Linear Regression employs a linear approach, seeking to minimize the sum of squared errors between observed and predicted values. This algorithm performs optimally when the data exhibits a linear relationship between the input features and the target variable. However, its simplicity may falter when confronted with more complex patterns or nonlinear relationships.

SVR, a powerful technique rooted in statistical learning theory, constructs an optimal hyperplane that best fits the data. Utilizing a user-defined margin, this approach enables the algorithm to capture both linear and nonlinear relationships by leveraging kernel functions. Despite its flexibility, SVR can be sensitive to hyperparameter selection and may require fine-tuning to achieve optimal performance.

kNN embodies an instance-based, lazy learning approach. By examining the k closest neighbors to a query point, it generates predictions based on the majority vote or average value of these neighbors. Although adept at handling nonlinearities, kNN may struggle with high-dimensional data and is sensitive to the choice of k, distance metric, and feature scaling.

ANN, inspired by biological neural networks, consists of interconnected nodes or neurons organized into layers. These networks learn intricate patterns and adapt to various data distributions through a process called backpropagation. While ANNs excel in modeling complex relationships, they can be computationally intensive and prone to overfitting without proper regularization and architecture design.

Discussing the Results

Although the results across models were quite similar, the SVR model emerged as the most accurate, boasting an RMSE of 0.2137, whereas the kNN model demonstrated the lowest accuracy. As evident from the plots above, none of the models achieved particularly high accuracy, especially in predicting the highest and lowest power generation values.

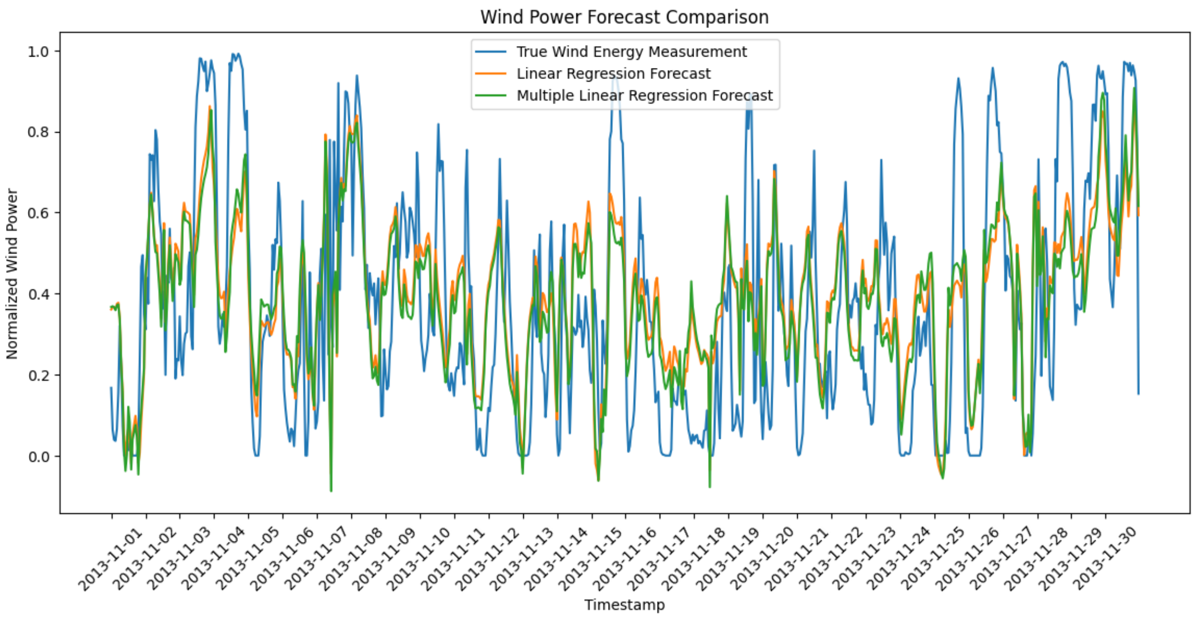
The data's scattered nature, which reveals a less apparent correlation between wind speed and wind power, contributes to this issue. Other factors also impact power generation output, complicating predictions based solely on wind speed. The ANN and SVR models' increased accuracy compared to linear regression can be attributed to the fact that when plotting wind generation against wind speed, the resulting shape aligns more closely with a curved line than a straight one.

The kNN model's low accuracy is also a result of the weak correlation between wind speed and wind power. Consequently, the closest wind speed from the training dataset may yield a power output significantly different from the predicted output.

Task 2

For task 2 we have created a time-series comparing the actual wind strength, the predicted wind strength using linear regression and predicted wind power using multiple linear regression.

Time-series Graph



Forecasting Accuracy (RMSE)

Model	**RMSE**
Linear Regression	0.2164
Multiple Linear Regression	0.2156

Does wind direction matter when predicting wind strength?

As we can see in the table above that compares the RMSE we can see that MLR has a slightly better accuracy than the Linear Regression model. This means that in some way the wind direction plays a role. However the difference is small and does not affect the results much overall.

Task 3

We used One-Step Ahead for task 3.

How the data (input and output) was encoded

In this task, we aim to explore the performance of different machine learning models, specifically Linear Regression, Support Vector Regression (SVR), Artificial Neural Networks (ANN), and Recurrent Neural Networks (RNN), in predicting power generation. A crucial aspect of developing these models is preparing the training data, which includes encoding the input and output data. Here, we use the sliding window method for this purpose.

The sliding window method involves creating a fixed-size window that slides over the time series data. At each step, the window extracts a sequence of data points as input and the subsequent data point as output. This approach allows the model to learn from the patterns in the data and make predictions based on a given sequence.

This prepared data can now be fed into the Linear Regression, SVR, ANN, and RNN models, allowing them to learn from the patterns and make predictions accordingly. It is essential to tailor the architecture and hyperparameters of each model to optimize their performance on the encoded input and output data.

Windowsize 1:
Input: $X = [[x_1], [x_2], ..., [x_n]]$
Output: $Y = [x_{1 + windowsize}, x_{2 + windowsize}, ..., x_n]$
Windowsize 2:
Input: $X = [[x_1, x_2], [x_2, x_3], ..., [x_{n-1}, x_n]]$
Output: $Y = [x_{1 + windowsize}, x_{2 + windowsize}, ..., x_n]$

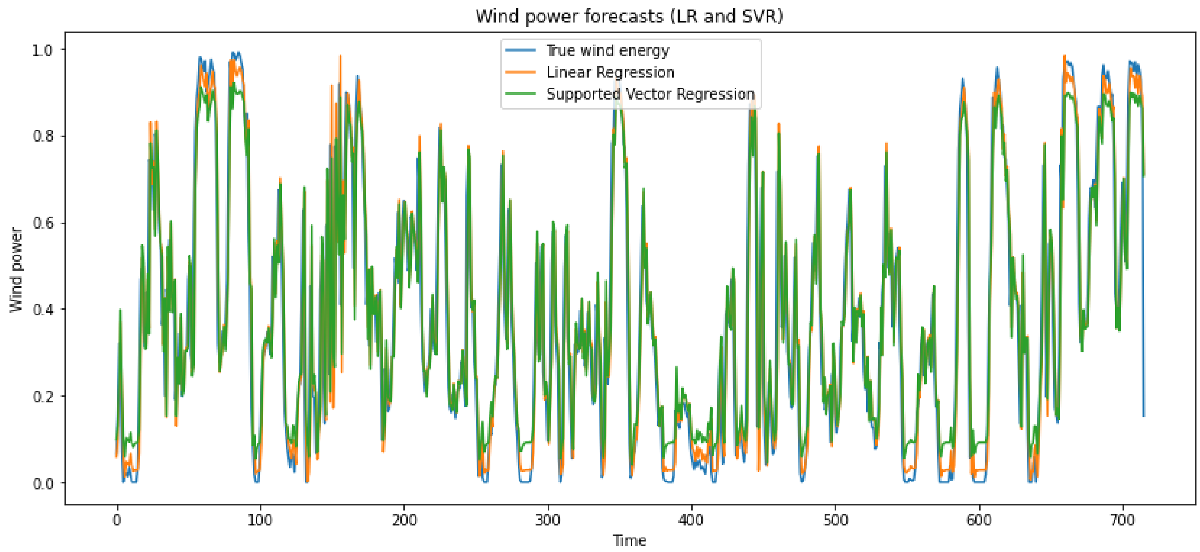
Implementation

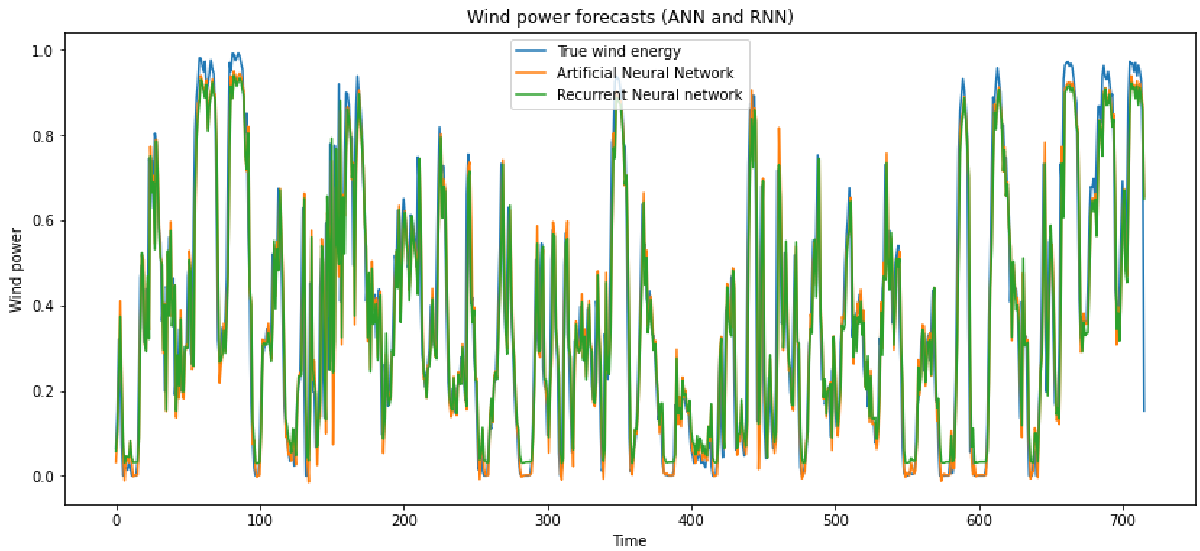
The prepare_data function is defined to preprocess the power generation data using the sliding window method. Next, the training data is read from the TrainData.csv file, and the actual power data for comparison is obtained from the Solution.csv file. The prepared input data (x) and output data (y) are then generated by passing the training power data through the prepare_data function.

Four distinct functions are defined for the implementation of the models: regression for Linear Regression and SVR, ann_model for ANN, and rnn_model for RNN. These functions take in the input and output data along with other necessary parameters, such as the window size and epochs, and return the Root Mean Squared Error (RMSE) and predicted power values.

After training and evaluating each model, the RMSE values are printed for comparison. The predicted power values are then saved in separate files using the provided ForecastTemplate.csv. Finally, the implementation concludes with plotting the time-series figures for the models, showcasing the wind power forecasts in comparison to the true wind energy data. A tabulated summary of the forecasting accuracy for each model, measured by their RMSE, is presented to facilitate an easy comparison of their performance.

Time-series Graph





Forecasting Accuracy (RMSE)

The results below show that ANN and RNN models are better than LR and SVR. And also has results for different window sizes. We also have a comparison of the

RMSE values with different windowsizes for ANN and RRN below. The most accurate model is RNN with windowsize 5. (For epochs 100)

RMSE → **Model ↓**	**winsize 2**	**winsize 4**	**winsize 6**
Linear Regression (LR)	0.1235	0.1233	0.1236
Support Linear Regression (SVR)	0.1236	0.1232	0.1235
Artificial Neural Network (ANN)	0.1196	0.11949	0.1227
Recurrent Neural Network (RNN)	0.1213	0.11953	0.1210

RMSE by windowsize - epochs 100

Blue - RNN
Orange - ANN