Feature Engineering and Model Evaluation on Wind Speed Time Series Dataset

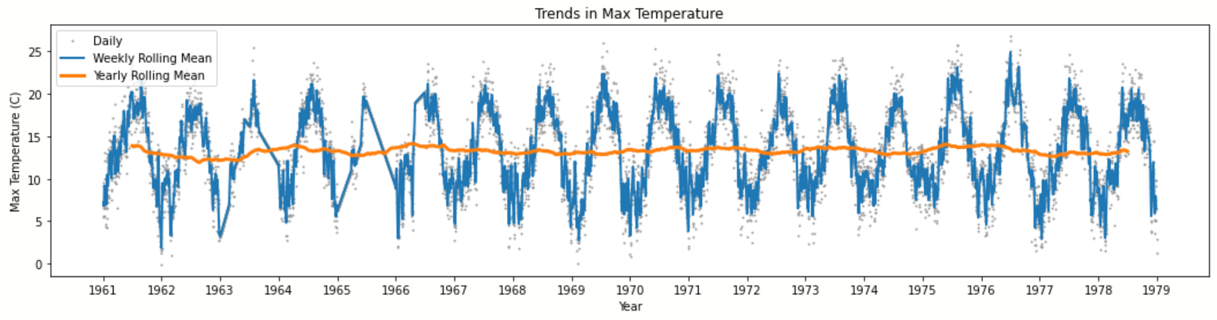
Juntae Park Reiden Webber

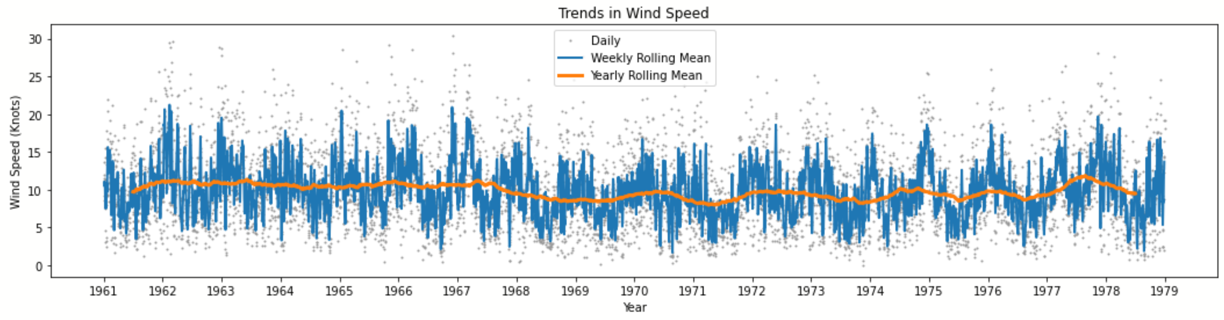
Introduction

Wind speed has much more impact than what we would think - the strong wind could cause natural disasters such as typhoons and tornados [1], while an ideal range of wind speed (which is above 9 miles per hour or above) can maximize wind power that produces renewable energy [2]. In the past, there has been some research on wind speed prediction using measurements from neighboring locations and combining the extreme learning machine and the AdaBoost algorithm [3]. Now, our goal is to train a model using a neural network by utilizing linear layers and ReLU functions that can predict wind speed based on multiple factors such as precipitation, maximum temperature, minimum temperature, and minimum grass temperature. Furthermore, we'll be comparing different models and algorithms to predict wind speed and manipulating datasets to increase our prediction's accuracy. Our data set includes 6574 days of average wind speed as well as those factors mentioned above. Eventually, once we have a model to predict wind speed, we could use it in various fields - for instance, we could possibly anticipate a big natural disaster coming, or find an ideal location to implement wind turbines.

Data

Our data was downloaded from Kaggle, which contains 6574 instances of daily averaged responses from an array of 5 weather variables sensors embedded in a meteorological station [4]. The device was located on the field in a significantly empty area, at 21M. Data were recorded for 17 years, starting from January 1961 to December 1978. In our code, we first started by reading the CSV file using *panda.read_csv* which reads CSV data into Dataframe. Out of 9 columns that were provided in the original CSV file, we decided to use 5 columns - "DATE"(YYYY-MM-DD), WIND(knots), "RAIN"((mm), "T.MAX" (°C), "T.MIN" (°C), "T.MIN.G" (°C). The reason we chose those columns is that the other 3 columns are indicator values, which we decided to omit as we weren't sure what the significance was of those three indicators. Since we're dealing with time-series data, we had our Date column represented as a Timestamp using *panda.DateTime* function. This helps to deal with the dataframe as its indices are no longer represented from 0 to 6573, but instead, it has a format of year-month-day format. For max temperature, min temperature columns, and grass temperature, we had a decent amount of NA values. We decided to use linear interpolation by calling the interpolation method from the pandas dataframe object. One of our main shifts from the earlier phase of codes into later was how we used our data to train a model. At first, we didn't look at the trend of the data and tried to manipulate the training data by using the inverse log function. Basically, our idea was just giving less weight to each data as it gets further away from our target date. However, our initial result in this way wasn't great, so we decided to do some more research and discover why it wasn't giving the result we expected. Then, we figured out that different seasons and different months had a big impact on the wind speed and other features. That was the main thing that we didn't consider from the beginning - finding the trend of the data, which is one of the most important features of time-series data. After realizing that, we've tried resampling by week and year. Below are some examples of visualization that show the trend of max temperature and wind speed.





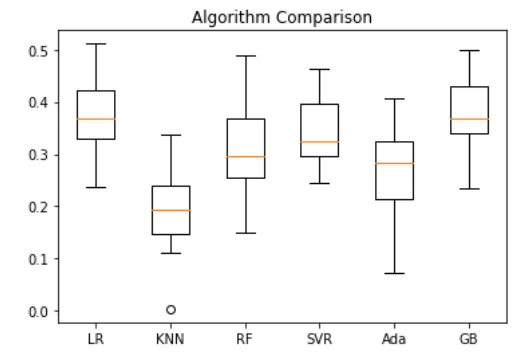
The orange line shows the yearly rolling mean, and the blue one shows the weekly rolling mean. We've only included two graph examples in this report (Max temp and wind), however as you can see in the codes, we've tried resampling for every feature. Visualizing those let us conclude that it'd be very important for us to consider the periodicity and trend of our data. Once we notice this trend, we decided to implement the feature engineering influenced by this article, which we'll incorporate more in detail under the method section [10]. Then, we used the StandardScaler method, which will normalize individual data. (Actually, we eventually mention it in the finding section, but for some reason, it made our result worse, therefore getting rid of it at the end). For training data selection, we chose the first 12 years of data (from 1961 to 1963) to be the training data and chose the 5 years of data to be our testing data (from 1964 to 1978). We followed a conventional 7:3 split between training data and testing data [8].

Methods

From the *Data* section of our report, we analyzed each of the features to find any trends within the data. We found that there was a weekly trend within both wind and temperature data that had a periodicity of 1 year. Therefore, we hypothesized that features that take into account data from previous days and weeks would lead to greater improvement in prediction.

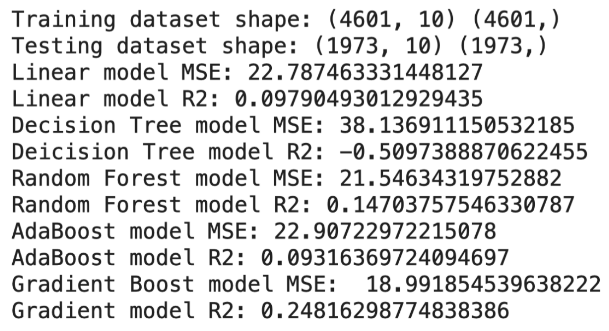
In our *Feature Engineering* section of the code, we originally had 4 features: "Max Temperature", "Minimum Temperature", "Minimum Ground Temperature", and "Rain Precipitation". Because the "Rain Precipitation" data had many outliers and a seemingly insignificant trend, we decided to not extract any other features from it. We used a technique where we included "lag features" which were features that represented the data from a certain number of days/weeks ago as well as the difference between that day and the current data point. The “lag features”, in theory, would allow the model to capture any seasonality or trends from the previous data. To accomplish this, we took advantage of the panda library’s *shift()* and *diff()* functions. We used the shift() function to shift the data by 1 day (to account for daily periodicity) and 7 days (to account for weekly periodicity), then we used the diff() function for the difference between the original and shifted days. We decided to use this technique to extract these types of features within time series because a similar group used this same technique among others [10] and it may help the model identify the trend we found in the *Data* section. After feature engineering, we produced 15 new features with which we could train.

Next, we wanted to cross-validate our features against different models. Following a similar method used in predicting crude oil prices [11], we decided to go against using standard k-fold cross validation and instead use a time series split cross-validation. In the standard version, there are elements of randomization which we wanted to avoid given our data deals with time-related information. Therefore, the time series split cross-validation technique would allow us to train on the first year’s data and then test on the second year, then train on the first two year’s data and test on the third year, etc. We decided to compare 7 models with our time series (the 7th one was not shown in the box plot). We chose “Linear Regression”, “K-Nearest Neighbors”, “SVR”, and “Neural Network” for our first four because they were common regression models that we went over in class. The other three, “Random Forest”, “AdaBoost”, and “Gradient Boost” are the topical models to use for time series. According to Zhang et. al., “Ensemble learning can achieve higher accuracy and robustness, as well as a better generalization ability than a single model in most problems. It does not strive for absolute accuracy in the construction of simple learners and can produce better results in small samples” [11]. Thus, we hypothesized that ensemble learning models would suit this time series better than the other models. For the cross-validation, we used the Sklearn libraries for the models and the metrics in evaluating the models. For the specific cross-validation method we used, we compared the means and standard deviation of the r2 values of each model. The results from the box plot above show us that Gradient Boosted Regression performed the highest on average, followed by Linear Regression and SVR.

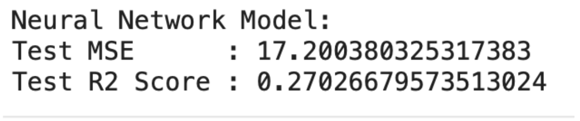


In most cases, we would proceed with the highest performing model and adjust to that model to achieve the closest predictions possible. Instead, we decided to test each model with the proper testing data that we created earlier with the results in the *Findings* section of this report. In the final section of our code, we defined a neural network from scratch and trained it with tensor data. We wanted to incorporate some of what we learned in class such as converting data into tensors, and adjusting the hyperparameters as we saw fit.

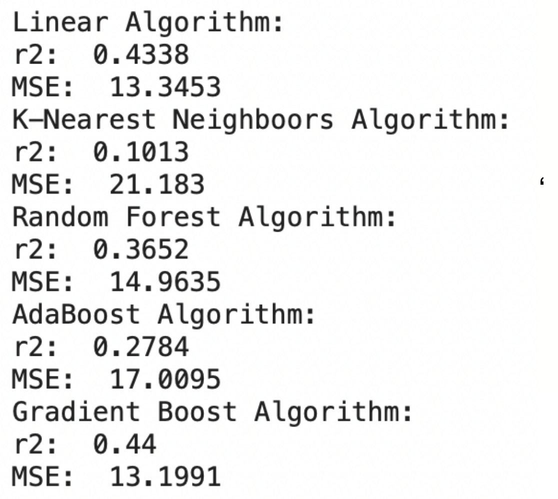
Findings



```
Training dataset shape: (4601, 10) (4601,)
Testing dataset shape: (1973, 10) (1973,)
Linear model MSE: 22.787463331448127
Linear model R2: 0.09790493012929435
Decision Tree model MSE: 38.136911150532185
Deicision Tree model R2: -0.5097388870622455
Random Forest model MSE: 21.54634319752882
Random Forest model R2: 0.14703757546330787
AdaBoost model MSE: 22.90722972215078
AdaBoost model R2: 0.09316369724094697
Gradient Boost model MSE:  18.991854539638222
Gradient model R2: 0.24816298774838386
```



```
Neural Network Model:
Test MSE      : 17.200380325317383
Test R2 Score : 0.27026679573513024
```



```
Linear Algorithm:
r2:  0.4338
MSE:  13.3453
K-Nearest Neighboors Algorithm:
r2:  0.1013
MSE:  21.183
Random Forest Algorithm:
r2:  0.3652
MSE:  14.9635
AdaBoost Algorithm:
r2:  0.2784
MSE:  17.0095
Gradient Boost Algorithm:
r2:  0.44
MSE:  13.1991
```

After all, although our end result of R2 value isn't as high as we expected it to be, we've seen a lot of improvement in our MSE value and R2 value. The image on the left top is the listed values of the R2 value and MSE value using the raw dataset without any data manipulation or featuring and simply splitting the dataset into 7 : 3 (training: testing ratio). The image on the left bottom is using neural networks that were influenced by projects 1 and 2. And the image on the right is the listed values of the R2 value and MSE value of various models after feature engineering throughout our codes. The reason we used the R2 value and MSE value is mainly that for regression-type models, we're trying to find how close our predicted value is to the true value. The smaller the MSE values are, the bigger the R2 values are, it would be closer to the ideal result. For some reason (we tried to look up why this was happening but couldn't really figure it out because I think it really depends on situations), scaling and normalizing data our training data set wasn't helping the result. When we got rid of the standard scaler feature that we mentioned in the data section, our performance was enhanced. We were quite surprised by how the AdaBoost model was actually performing pretty badly compared to Linear Regression and Random Forest model after our data featured compared before especially because there was a research paper that utilized the AdaBoost model to predict wind speed [3]. In gradient boosting machines which had the best result out of all the models we've tested, the learning procedure consecutively fits new models to provide a more accurate estimate of the response variable. The principle idea behind this algorithm is to construct the new base-learners to be maximally correlated with the negative gradient of the loss function, associated with the whole ensemble [6]. We were inspired to try gradient based because we read an article that mentioned that "XGBoost is the best performing model out of the three tree-based ensemble algorithms and is more robust against overfitting and noise." [7] in time-series datasets. Comparing our new data featuring to the old one (basically going from log inverse to feature engineering), I think our result had notable improvements as we gave more information to our model about a day before and a week before as well as the periodicity. This is also known as differencing, which was a big shift from the earlier phase of a project to the final phase. Looking at the original result of R2 values, we believe the wind data itself was weakly correlated to other features such as temperature and rain in the first place, but by manipulating the data and with feature engineering we were able to observe a somewhat moderate amount of correlation.

Future Work

One of the main challenges that we faced was learning how to deal with time-series data as it was a new concept for us. We notice that seasonal variation exists (which prevented us from just randomizing the training data) and tried to essentially remove this trend by using differencing [9] and measuring how much *different* it is from the value in the preceding point in time instead of using the actual value at the particular point in time. Because we spent lots of time understanding this idea, and due to the limited amount of time we had, we didn't have enough time to look into different models and algorithms and find the most optimal hyperparameters for each model. Also, as we mentioned earlier, we're still not sure why the normalization process was worsening the models' performance, so it's one of the things we could be working on. Other things we could've done were to include the resampling mean as one of the features and see if that would've improved the model, separate weeks and year numbers as features, or try to fill up null values with other methods instead of interpolating such as forward filling, backward filling, and nearest neighbor. What we found though, is that by using four main features from our dataset, we can get a model that can somewhat predict wind speed based on those features. Although the R2 value isn't as high as we wanted it to be, approximately half of the observed variation can be explained by the model's inputs (I'm simply interpreting the R2 value in a statistical way, and trying to be careful because we realize a minor wording can mean a whole different thing in statistics). In the long term, if we implement the possible improvements and get a better dataset that has a somewhat higher correlation, one of the real-world applications would be (as we mentioned in the introduction) improving the current severe weather prediction algorithm. If we can advance our current weather warning system, the government and citizens would have a better time preparing for it. Another possible implementation would be using it for wind energy utilization, as wind energy utilization is a popular, sustainable, renewable energy source that has a much smaller impact on the environment than burning fossil fuels. Being able to predict the windspeed will help us to find ideal spots to build wind generators and maximize the energy that will be collected.

References

[1] Statistical Summary and Case Studies of Strong Wind Damage in China. https://www.researchgate.net/publication/289575580_Statistical_Summary_and_Case_Studies_of_Strong_Wind_Damage_in_China.

[2] “U.S. Energy Information Administration - EIA - Independent Statistics and Analysis.” Where Wind Power Is Harnessed - U.S. Energy Information Administration (EIA), https://www.eia.gov/energyexplained/wind/where-wind-power-is-harnessed.php.

[3] Wang, Lili, et al. “Wind Speed Prediction Using Measurements from Neighboring Locations and Combining the Extreme Learning Machine and the ADABOOST Algorithm.” Energy Reports, Elsevier, 6 Jan. 2022, https://www.sciencedirect.com/science/article/pii/S2352484721015018.

[4] fedesoriano. (April 2022). Wind Speed Prediction Dataset. Retrieved [Date Retrieved] from https://www.kaggle.com/datasets/fedesoriano/wind-speed-prediction-dataset

[5] Trovero, M.A., & Leonard, M.J. (2018). SAS 2020-2018 Time Series Feature Extraction. https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/2020-2018.pdf

[6] Natekin, Alexey, and Alois Knoll. “Gradient Boosting Machines, a Tutorial.” *Frontiers*, Frontiers, 1 Jan. 1AD, https://www.frontiersin.org/articles/10.3389/fnbot.2013.00021/full.

[7] Nikulski, Julia. “Time Series Forecasting with AdaBoost, Random Forests and Xgboost.” *Medium*, Towards Data Science, 29 Sept. 2021, https://towardsdatascience.com/go-highly-accurate-or-go-home-61828afb0b13.

[8] Kreinovich, Vladik, et al. “Spatial Resolution for Processing Seismic Data: Type-2 Methods for Finding the Relevant Granular Structure.” *ScholarWorks@UTEP*, https://scholarworks.utep.edu/cs_techrep/12/.

[9] “Forecasting: Principles and Practice (2nd Ed).” *8.1 Stationarity and Differencing*, https://otexts.com/fpp2/stationarity.html.

[10] H. Assaad, Rayan & Fayek, Sara. (2021). Predicting the Price of Crude Oil and its Fluctuations Using Computational Econometrics: Deep Learning, LSTM, and Convolutional Neural Networks. Econometric Research in Finance. 6. 119-137. 10.2478/erfin-2021-0006.

[11] Zhang, Y, Ren, G, Liu, X, Gao, G, Zhu, M. Ensemble learning-based modeling and short-term forecasting algorithm for time series with small sample. Engineering Reports. 2022; 4(5):e12486.