PSDS Capstone Final

Capstone Goal

Determine if any one, or a combination of several, weather data sources can accurately predict the wind speed in my backyard.

Importance

Currently, the Homeowners Association where I live does not have architectural standards for personal wind-electric generation equipment. I hope to produce a model to retroactively predict an approximation of wind speed in my backyard and then use the model to estimate roughly how much electricity I might have produced within a given past time range. I hope to eventually use this prediction technique to develop a comprehensive model for wind-electric potential for my entire community and use that data to propose architectural standards for personal wind-electric generation equipment.

Data Sources

Data Sources Used

- My personal weather station (Ambient Weather WS-2902C)
- Weather Underground - Weather Underground (www.wunderground.com) provides a robust, well documented API that allows queries for historical weather readings and the ability to obtain historical data from any weather station that contributes data to their repository.

Data Sources Evaluated But Not Used

- Openweather Map - Openweather Map (openweathermap.org) provides readings for the same times and locations as Weather Underground stations, but the data is formatted very differently. Further, the values provided by Openweather Map do not correspond to readings for the same time and location as personal weather stations found on Weather Underground.
- Ambient Weather - Ambient Weather (ambientweather.net) is the native repository for readings from my weather station. However, it provides no ability to access data from weather stations other than my own. Also, the ability to obtain historic data from my weather station is extremely limited.
- Weathercloud - Weathercloud (weathercloud.net) provides no API and no ability to access data from weather stations other than my own.

Data Sources Not Evaluated

- Citizen Weather Observer Program (CWOP/APRSWXNET) via Meteorological Assimilation Data Ingest System (MADIS) at NOAA - CWOP does provide API access,

but only via a special Fortran program or via helper utilities that one must download and install. The time demands of the capstone precluded a more complete investigation of this data set.

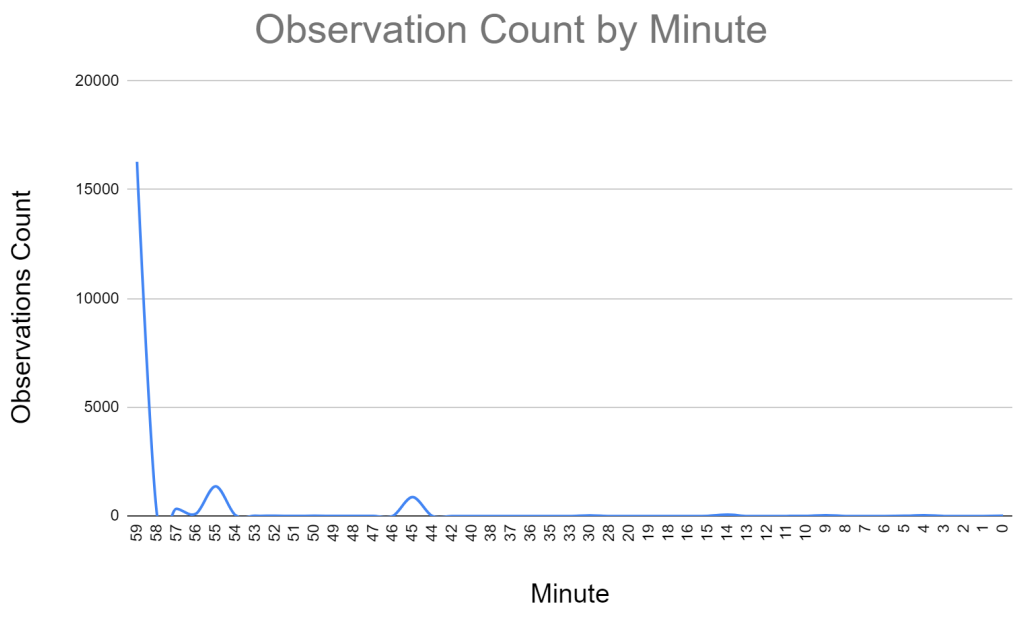
Local Data Storage

Due to Weather Underground API limitations, it was necessary to create a local database to store observations. To store the data I created a local MariaDB database running on a Raspberry Pi attached to my local network The database consists of two tables; the first table, `wu_station`, contains station information such as station identifier (stationID) and coordinates. On insert, the lat/lon values are used to construct a POINT tuple for use in follow-on geospatial queries. The second table, `wu_observations`, stores the observations. An entity relationship diagram of the tables can be found in the Weather Underground ERD and data formats can be found in the Data Dictionary.

Identifying Time Groupings

Although each weather station typically reports readings hourly, they do so on their own schedule. The vast majority of stations in the data report readings at 59 minutes past the hour but a significant amount of stations report at other times (see Figure 1).

Figure 1: Observation Count by Minute



In order to compare observations across weather stations I could not simply index by raw timestamp. I needed to identify the best way to group observations by time that maximized representation of all weather stations within each group and ensured that my weather station

was present in as many groups as possible. I tested time groupings of 5, 10, 15, 20, 30, 45 and 60 minutes. Table 1 shows the presence statistics for my station within each grouping.

Table 1: Presence of My Weather Station Within Each Grouping

	5min	10min	15min	20min	30min	45min	60min
My Station Present	266	265	265	264	263	263	263
Total Groups	880	590	577	540	314	351	263
Percent Groups Containing My Station	30.23%	44.92%	45.93%	48.89%	83.76%	74.93%	100.00%

If we round to the nearest 30 minutes 83.76% of the groups contain my station which was significant, but rounding to the nearest 60 minutes resulted in 100% representation of my station within each group. Groupings of 60 minutes were therefore used for analysis.

Data Ingest and Cleaning

As I expanded the number of stations and gathered more observations I had to devise a robust data cleaning and alignment process. Some weather stations appear to report measurements twice per hour which led to overrepresentation of those stations within the data set.

The data cleaning process uses a time delta component that identifies a time offset within each hour (by minute) that separates each station's readings from the target station. If a station is overrepresented within an hour group, only the reading with the smallest delta is retained. Figure 2 shows the raw number of measurements grouped by hour prior to cleaning and Figure 3 shows the per hour counts after cleaning. After cleaning, peaks have been reduced to near the optimal number of observations per hour (as represented by the red dashed line).

Also of note, during the course of the project a few stations stopped reporting readings. This accounts for the departure from the original 114 stations denoted by the red dashed line. Drops in station count represent hours for which data is missing from the repository and could not be recovered. There are numerous possible explanations for missing data such as internet or power outages and personal weather stations do not typically buffer readings to send when connectivity is reestablished.

Figure 2: Raw Number of Measurements by Hour

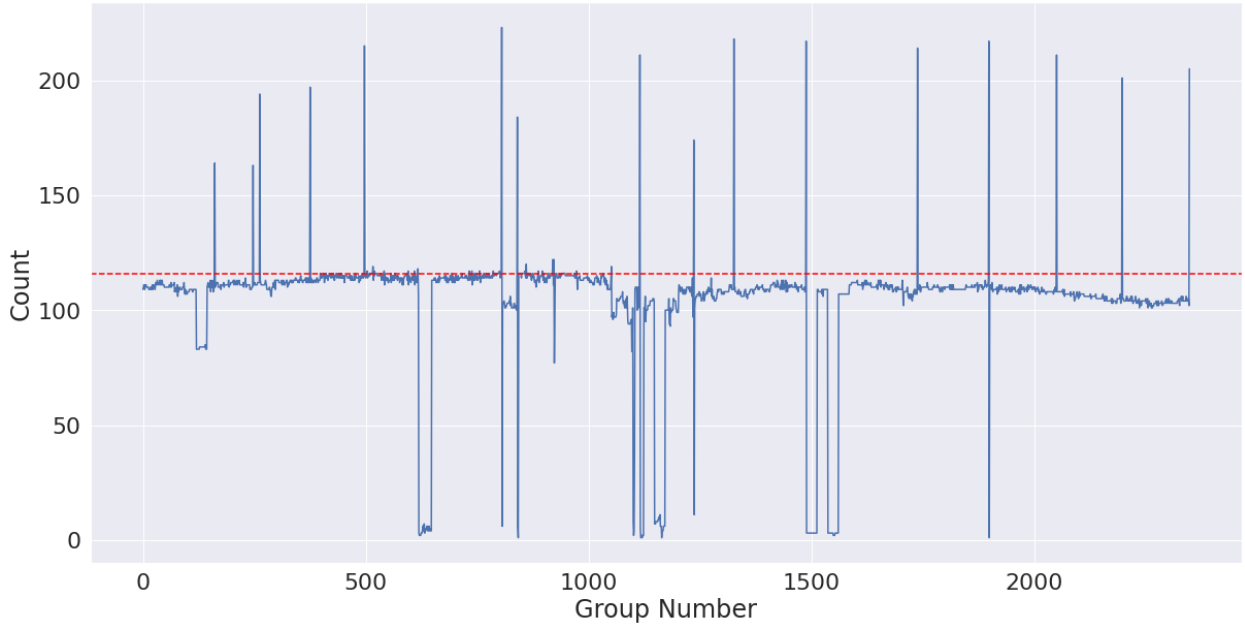
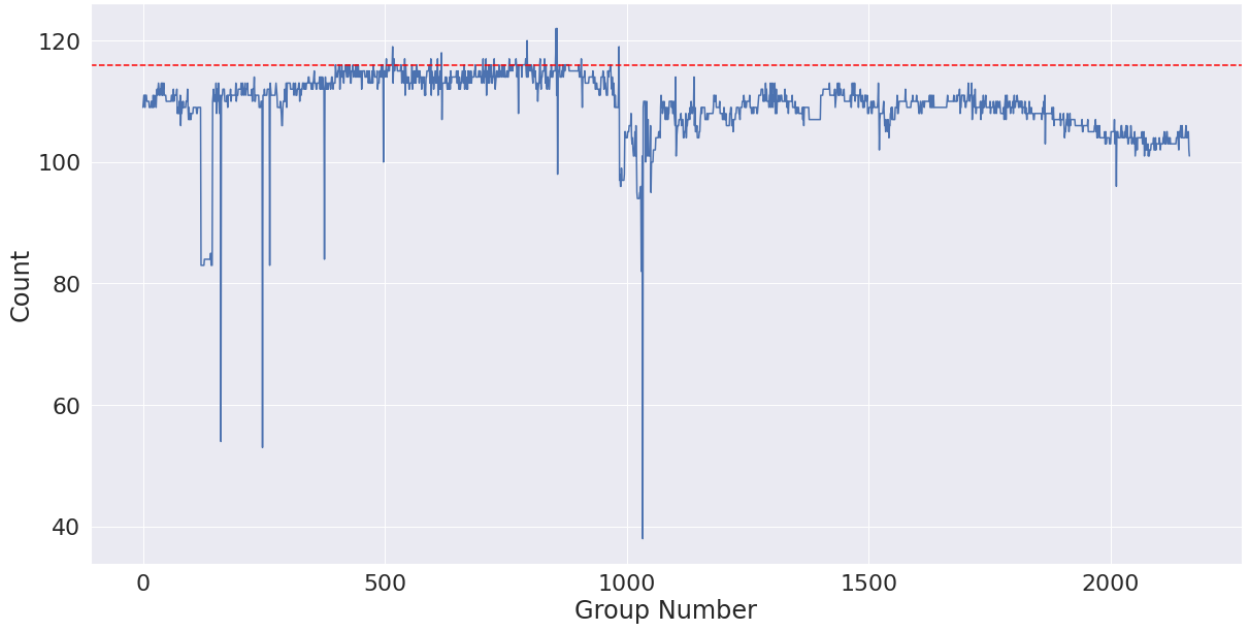


Figure 3: Measurements by Hour After Cleaning and Deduplication



Choosing Model Input

My first round of model experimentation only included windspeedAvg reading from N number of stations, where N is the number of stations that provided the best average accuracy (using KFold cross validation with K = 3) using a simple Linear Regression model and $1 \leq N \leq 30$.

Model accuracy based solely on windspeedAvg for N stations was unsatisfactory. After outlier elimination and scaling did not significantly improve scores, I devised a system to bring in more training data. First, I cross correlated all observations from the target station. I then used the correlation matrix to identify other variables highly correlated with windspeedAvg. Table 2 shows the top 10 variables most correlated with windspeedAvg within the target station's readings. Correlation scores significantly drop off after 0.80 so I decided to use that value as the natural cutoff. Using the cutoff value of 0.8 the variables windgustAvg, windspeedHigh and windgustHigh were added to windspeedAvg as additional model inputs.

Table 2: Correlation Scores for windspeedAvg within My Station's Data (truncated)

Variable	Correlation Score
windspeedAvg	1
windgustAvg	0.995071
windspeedHigh	0.852027
windgustHigh	0.838374
windgustLow	0.523414
windspeedLow	0.459209
solarRadiationHigh	0.370677
uvHigh	0.364316
tempHigh	0.286263
windchillHigh	0.28625

For all subsequent training and testing, the variables windspeedAvg, windgustAvg, windspeedHigh, and windgustHigh of the N most correlated stations were used.

Exploration of Data Distribution and Density

I next inspected the distribution of the values for windspeedAvg for the entire data set (Figure 4) and the N most correlated stations (Figure 5). The distributions for both the entire data set as well as the top fifteen most correlated stations are similar with the vast majority of values between zero and five miles per hour, tapering off to large outlier speeds.

Figure 4: Histogram of windspeedAvg Measurements for the Entire Data Set

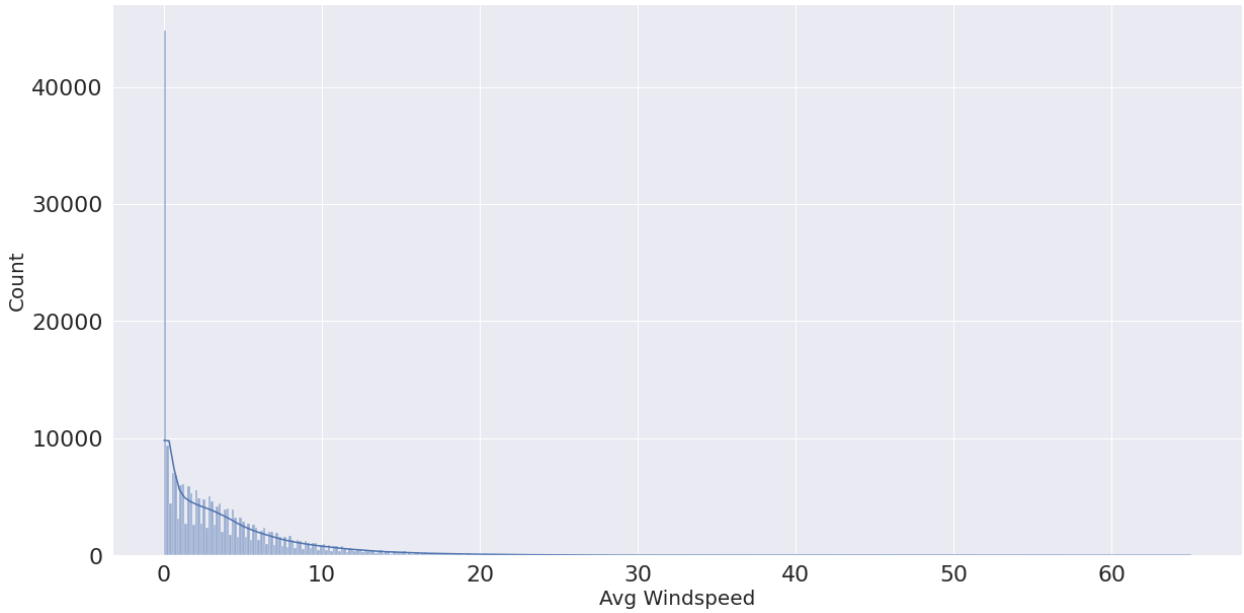
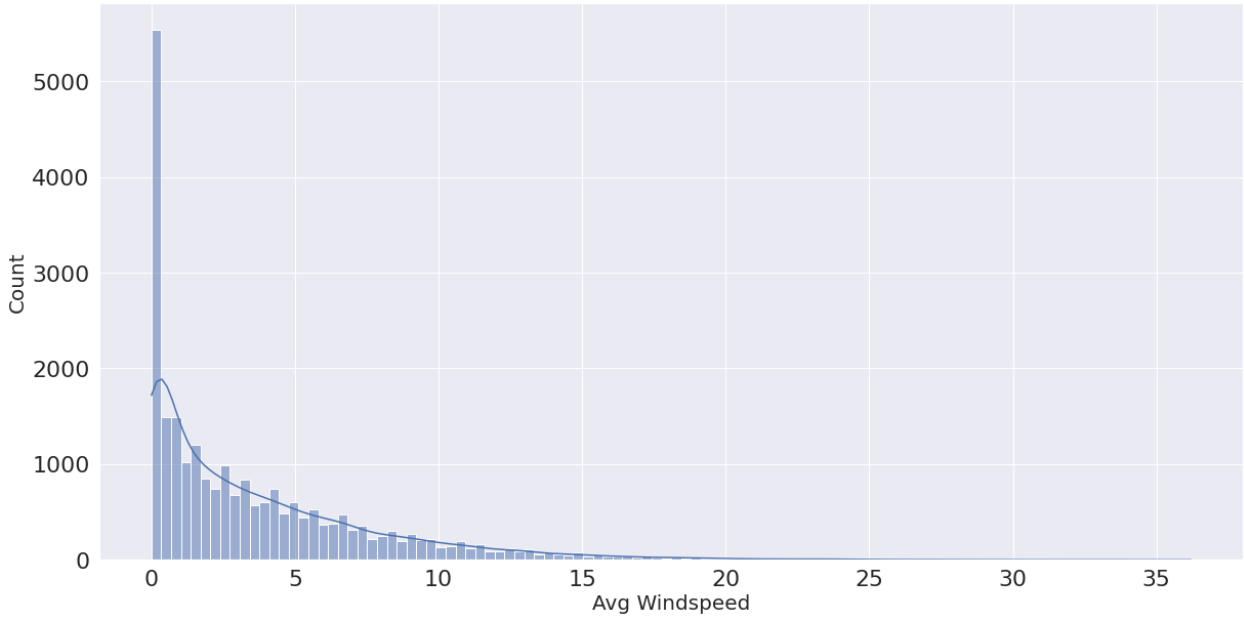


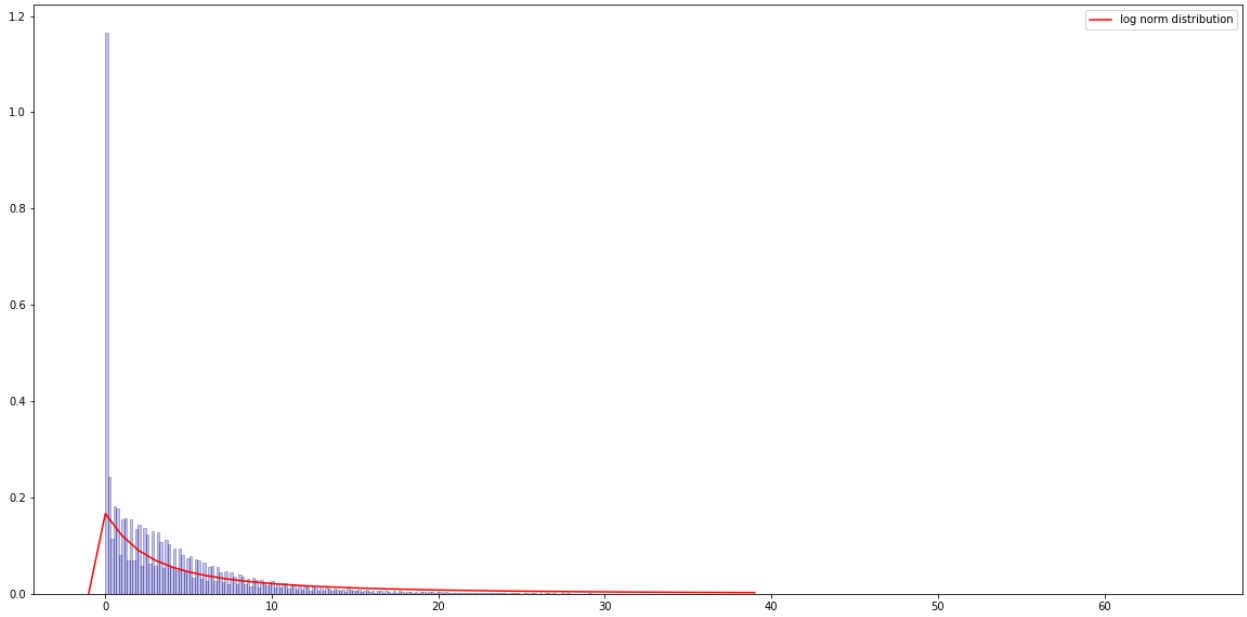
Figure 5: Histogram of windspeedAvg Measurements for Highly Correlated Stations



After some research, it appeared that this is possibly a log normal distribution. Overlaying the log normal distribution for the entire data set (Figure 6) we can see that it does look like a good fit. While researching the implications of modeling data that have a log norm distribution, I found

an excellent series of papers from the University of Gothenburg[1] advocating the use of Generalized Linear Models to accurately model the health impact of exposure to air pollution. This was the impetus for experimenting with the Tweedie and Poisson Generalized Linear Models.

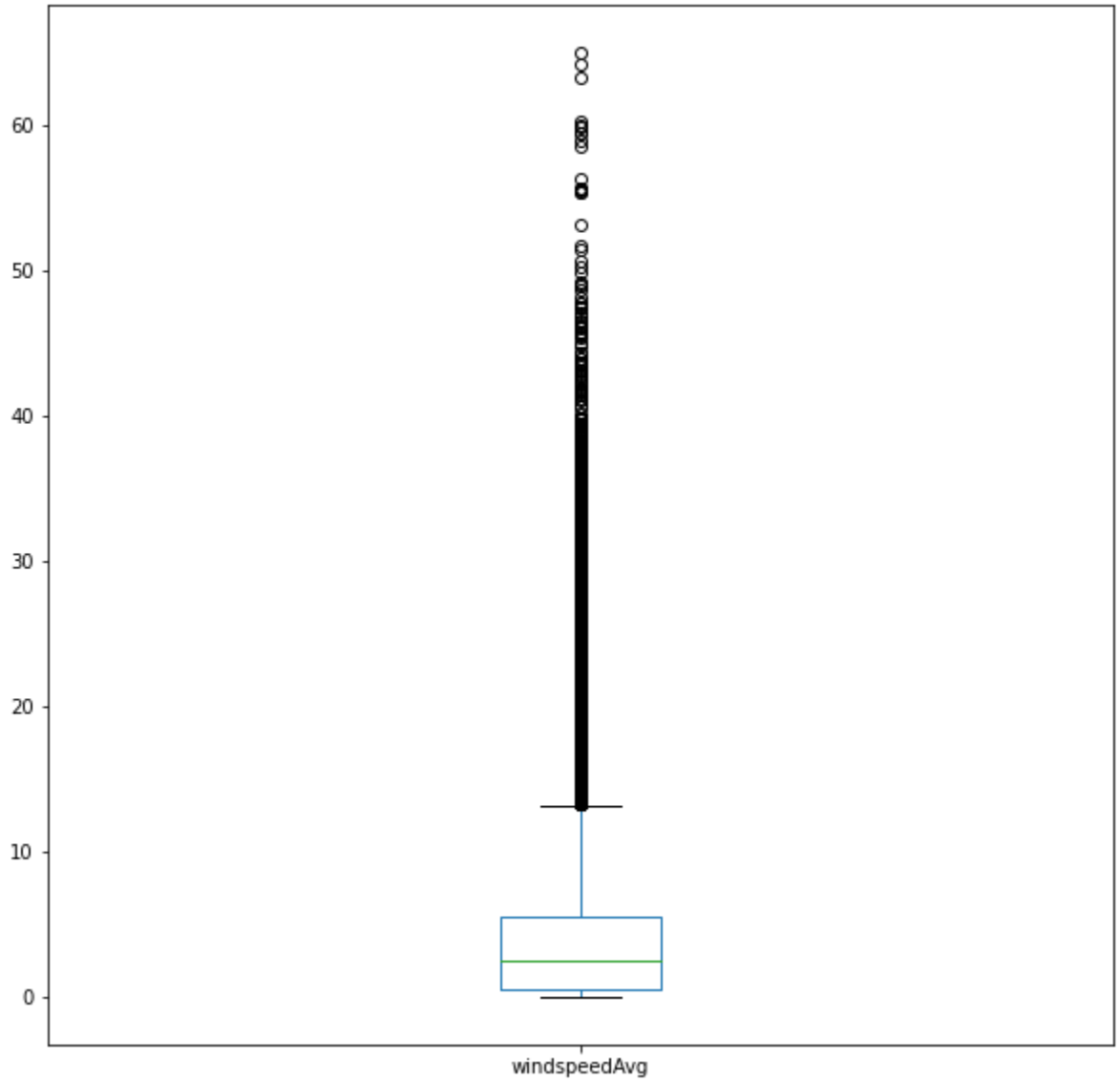
Figure 6: Histogram of windspeedAvg With Log Normal Distribution Overlay



After identifying the probable distribution of the data, I then analyzed the spread of the target variable windspeedAvg. The mean and first three quartiles of the data are fairly low and close together, but there are a *significant* number of outliers as evidenced by the boxplot in Figure 7.

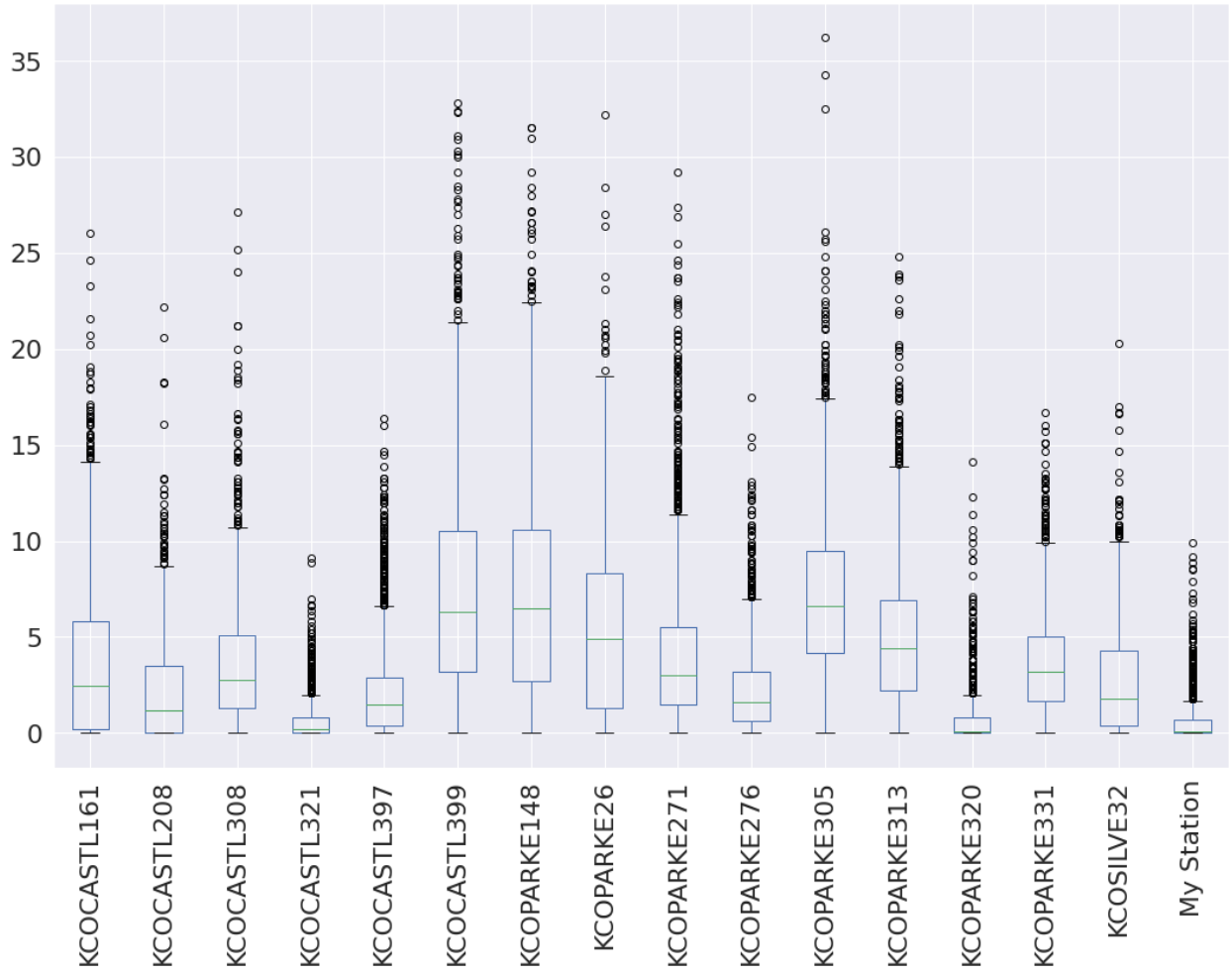
[1] Gustavsson, 2015 see Resources

Figure 7: Boxplot of windspeedAvg for the Entire Data Set



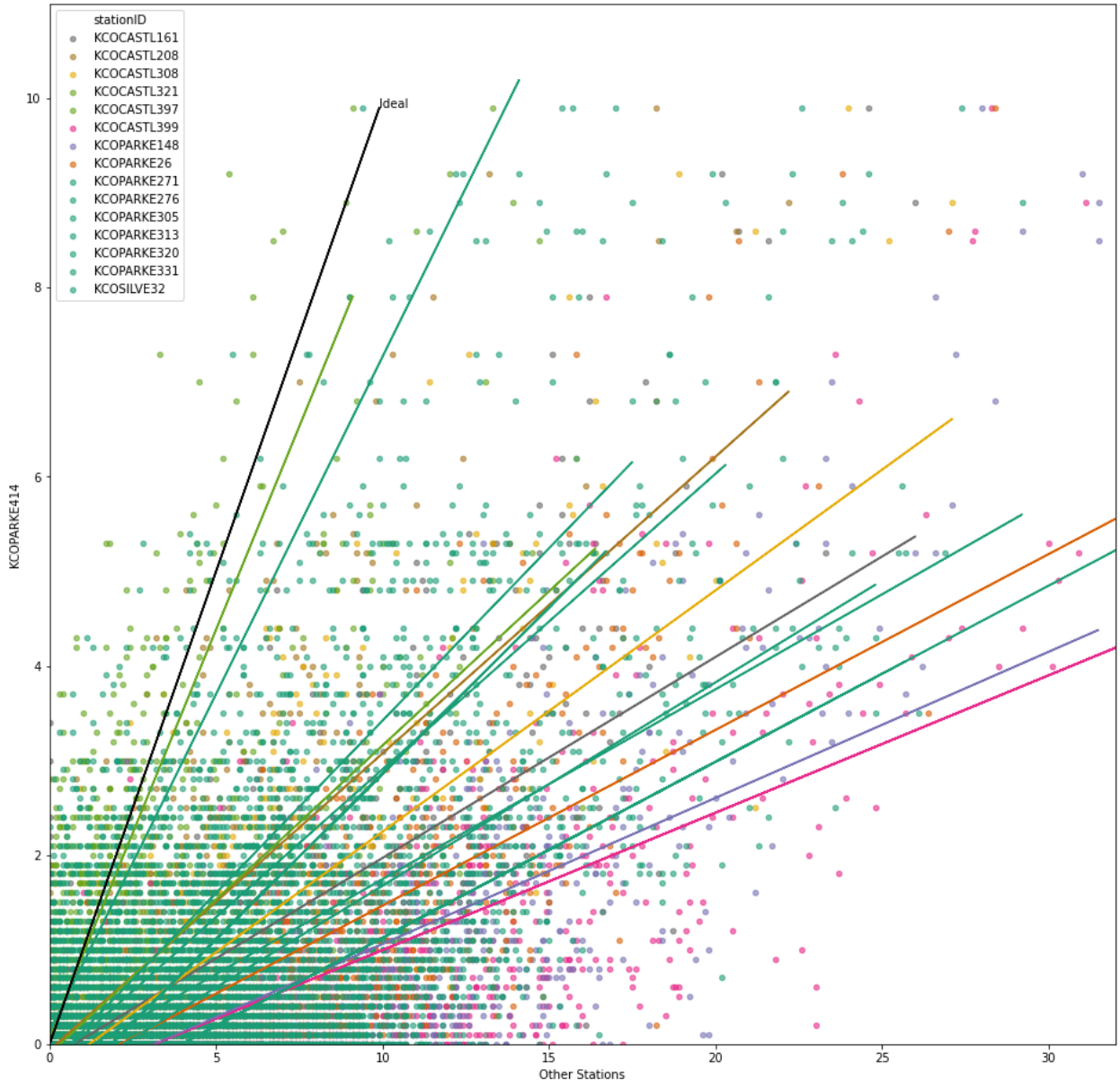
Examining the boxplots for the fifteen stations with windspeedAvg most highly correlated to that of the target (Figure 8) we see that some of the worst outliers from Figure 9 are not represented, but there are still a significant number.

Figure 8: Boxplot of windspeedAvg for Highly Correlated Stations



Plotting regression lines for the top most correlated stations in comparison to the target station (Figure 9) we see that stations KCOPARKE320 and KCOCASTL397 are the closest to to the target (denoted as “Ideal”), but all stations have flatter slopes due to the pull of their outliers.

Figure 9: Regression Lines for Highly Correlated Stations



Outlier Elimination

In an attempt to improve model accuracy, I experimented with removing outliers using the Isolation Forest and Local Outlier Factor methods. Removing outliers could potentially increase the slope of the correlated stations to more closely correspond to that of the target. Table 3 shows a side by side comparison of the effects of removing the outliers on the accuracy scores of four different models.

Table 3: Model Accuracy with Outliers Removed

	Entire Data Set	**Outliers Removed via Isolation Forest**	**Outliers Removed via LocalOutlierFactor**
Rows Dropped	0	159 (13.27%)	31 (2.59%)
Linear Regression Accuracy	0.873127	0.638218 (-27%)	0.680896 (-22%)
Ridge Regression Accuracy	0.766903	0.620214 (-19%)	0.739091 (-4%)
Lasso Regression Accuracy	0.768726	0.610656 (-20%)	0.70588 (-8%)
SVR (Poly degree 2) Accuracy	0.766903	0.697551 (-9%)	0.746221 (-3%)

The Isolation Forest approach identified 159 outliers, or 13.27% of the total observations. After removing Isolation Forest outliers and retraining, every model performed *significantly* worse, with model accuracy dropping between 9% and 27%.

The Local Outlier Factor approach identified 31 outliers, or 2.59% of the data. Of the four test models, Linear Regression performed significantly worse while State Vector, Ridge and Lasso Regression performed mildly worse.

Figure 10 shows the negative impact of removing outliers via Isolation Forest. After outliers are removed all of the stations are *less* correlated with the target. Figure 11 shows the impact of removing outliers via Local Outlier Factor. Fewer observations were removed using Local Outlier Factor which resulted in less impact, as evidenced in Table 3.

Figure 10: Regression Lines After Outlier Removal via Isolation Forest

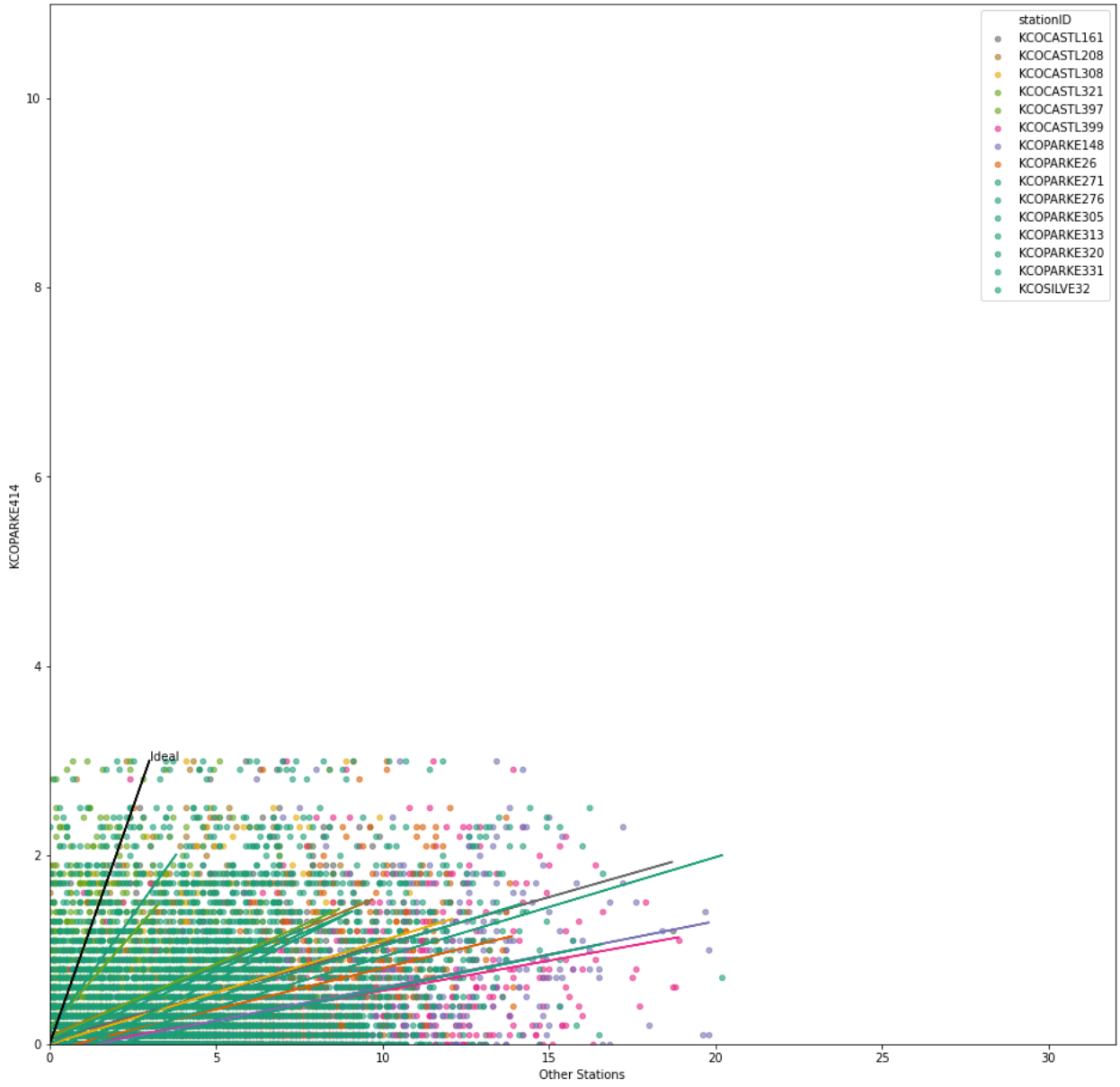
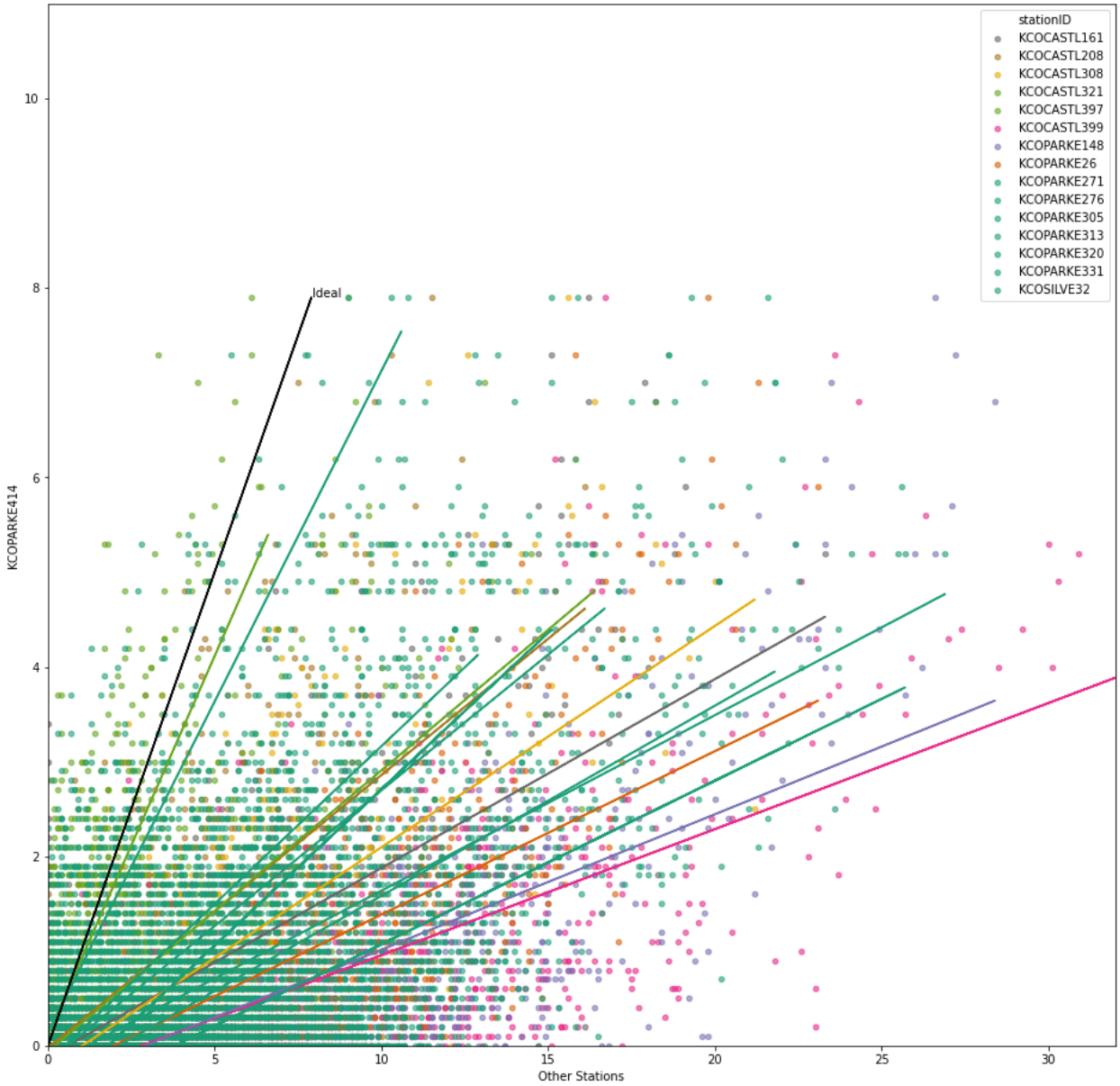


Figure 11: Regression Lines After Outlier Removal via Local Outlier Factor



The obvious importance of outliers led me to explore Generalized Least Squares models. Specifically, I experimented with a Huber Regression model to try and compensate for the apparent heteroscedasticity of the data.

Scaling Input with StandardScaler

After outlier removal did not improve accuracy scores, I next applied a StandardScaler to my Linear Regression, Poisson and Huber regressor pipelines. Scaling the input did not improve the accuracy of the Linear Regression or Poisson models, but significantly improved the accuracy of the Huber model from 0.74 to 0.80. The accuracy gain for the Huber model, however, was not enough to outperform the plain Linear Regression model.

Exploration of Prediction Models

A total of 10 models were evaluated using a train-test split of 85% training data and 15% testing data. The total amount of data available for training and testing was 1198 observations of 61 variables from 27 different stations. Model hyperparameters were tuned via Scikit-learn's Pipeline and GridSearchCV and best model parameters were used for evaluation. Table 4 lists each model and any variation along with the best hyperparameters found. The accuracy scores are the mean of the five scores obtained using five-fold cross validation.

Table 4: Model Testing Results

Model	Best Hyperparameters	Best Avg Accuracy
LinearRegression	N/A	0.873127
ElasticNet	'alpha': 0.01, 'l1_ratio': 1.0, 'max_iter': 2000, 'normalize': False	0.83345688
HuberRegressor with StandardScaler	'epsilon': 2.3, 'max_iter':201	0.803334
Lasso	'alpha': 0.01	0.768726
SVR poly kernel degree 2	N/A	0.766903
Ridge	'alpha': 0.01, 'normalize': True	0.765682
SVR poly kernel 2 localoutlierfactor removed	N/A	0.746221
HuberRegressor	'epsilon': 2.1, 'max_iter': 151	0.741381
Ridge localoutlierfactor removed	'alpha': 0.02, 'normalize': True	0.739091
LinearRegression with StandardScaler	N/A	0.738264
BayesianRidge	'alpha_1': 0.09, 'alpha_2': 0.01, 'normalize': True	0.706051

Lasso localoutlierfactor removed	'alpha': 0.02	0.70588
SVR poly kernel 2 iso outliers removed	N/A	0.697551
LinearRegression localoutlierfactor removed	N/A	0.680896
Poisson	'alpha': 1.8	0.654925
Linear Regression iso outliers removed	N/A	0.638218
Ridge iso outliers removed	'alpha': 0.02, 'normalize': True	0.620214
Lasso iso outliers removed	'alpha': 0.01	0.610656
RandomForestRegressor	'max_depth': 100, 'max_features':'auto', 'n_estimators': 50	0.582931
Poisson localoutlierfactor removed	‘alpha': 1.8	0.476733
Poisson with StandardScaler	‘alpha': 0.2	0.438463
Tweedie	‘alpha': 1.9 ‘power’': 1	0.390051

Predicting January

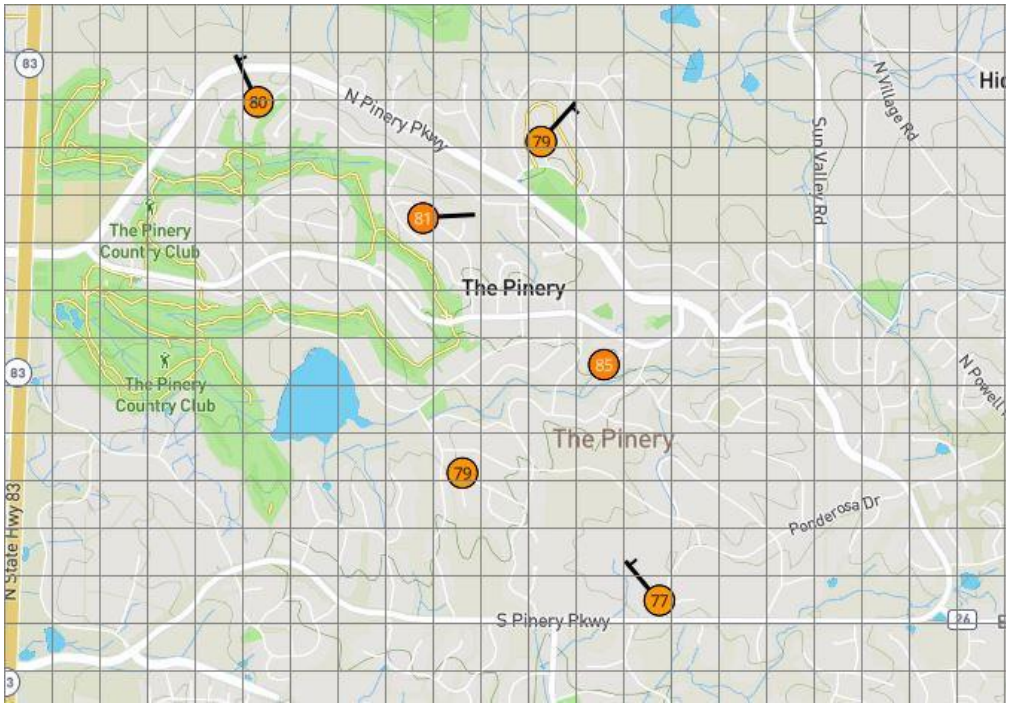
Based on the modeling exploration above, I elected to use the Linear Regression model to predict past average hourly wind speed for the target station. Training data from January 2021 were obtained from Weather Underground for the N most correlated stations. After data cleaning, alignment and deduplication the data set consisted of 580 hour groupings. The model predicted hourly average wind speed for the target station for the month of January at 0.439 miles per hour. Given the 87% average accuracy score of the model, expected actual average hourly wind speed should have been in the range 0.5 and 0.38 (+/- .13).

Next Steps

All code used for this project was purposefully designed to allow for application of the same modeling workflow for any other target weather station by simply changing a configuration file. The community in which I live currently has five other personal weather stations available via

Weather Underground. These stations can be processed and models developed for use in creating an interpolation grid for average hourly wind speed for the entire community. This would provide insight into probable wind speeds for areas without a weather station. Figure 12 is an example interpolation grid with 150 meter polygons with the locations of all the weather stations in my community. Ultimately, this data could be used to provide insight into creating and adopting architectural standards for personal wind electric generation equipment.

Figure 12: Example Grid Overlay of Community



Resources

All code is available via the project GitHub: https://github.com/riverdogcabin/PSDS4900

Gustavsson, Sara. University of Gothenburg, “Evaluation of Regression Methods for Log-Normal Data Linear Models for Environmental Exposure and Biomarker Outcomes,” Gothenburg Institute of Occupational and Environmental Medicine, ISBN 978-91-628-9295-1, 2015. [Online]. Available: https://gupea.ub.gu.se/bitstream/2077/37537/4/gupea_2077_37537_4.pdf. [Last Accessed: Aug. 12, 2021].

Data Dictionary

Weather Underground:

- Station Information
    - stationID: String
    - neighborhood: String
    - softwareType: String
    - country: String
    - lon: Float
    - lat: Float
    - elev: Float
    - station address: String
- Measurement Data (all measurement in metric unless otherwise noted)
    - obsTimeUtc: DateTime (YYYY-MM-DDThh:mm::ssZ)
    - obsTimeLocal: DateTime (YYYY-MM-DD hh:mm:ss)
    - solarRadiation: Float
    - realtimeFrequency: None
    - epoch: Integer
    - uv: Float
    - winddir: Integer
    - humidity: Float
    - qcStatus: Integer
    - temp: Float
    - heatIndex: Float
    - dewpt: Float
    - windChill: Float
    - windSpeed: Float
    - windGust: Float
    - pressure: Float
    - precipRate: Float
    - precipTotal: Float
    - units: String (“Metric”|”Imperial”)

Weather Underground API specifications can be found here.

Weather Underground ERD