Max LoGalbo
Dr. Joseph Antonelli
STA 4241
20 April 2022

Final Project Writeup

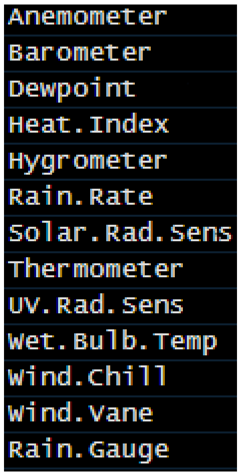
Introduction:

Often, when it rains on campus, I typically come poorly equipped, without an umbrella. The reasons for this come from a few reasons, mainly my poor preparation as well as poor reporting of rainy conditions on the weather app. The goal of this project is to solve the latter of the two issues. I would like to create a model that can accurately predict if I should be prepared for rainy weather. For the following "Cleaning & Manipulating Data" section, rainy conditions entail it had rained earlier that day. I decided, in addition, to set how far in the future to predict to 4 levels: 6 hours, 12 hours, 24 hours, and 48 hours into the future.

After some quick exploration of a few data sources, I found this source: https://alachua.weatherstem.com/data. Essentially, it contains climate sensor data for the last at least 7 years with various covariates instantiated, such as Anemometer, Barometer, and Dewpoint level. I ended up selecting 12 covariates from a diversity of options to predict rainy conditions. Overall, the final dimensions of the data I began to use totaled ~43000 rows or observations, with 13 covariates and the response variable.

Cleaning & Manipulating Data:

First and foremost, I had to create my response variable, a Boolean which dictates if it had rained earlier. The reason I chose the response to be if it had rained earlier is dependent on how the data is presented. Two variables in the data set, Rain.Gauge (in) and Rain.Rate (in/hr), are very similar. Rain.Gauge entails the amount of rain that had fallen that day. Rain.Rate entails the rate rain fell over the course of the past hour. Oftentimes within the data, Rain.Rate is given the value 0 and Rain.Gauge is given a non-zero value, contradicting one another. As a result, instead of setting the response variable to if Rain.Rate > 0 (to predict actual rainfall), I set the response variable to if Rain.Gauge > 0, as there simply exists a better likelihood of catching successes where it had rained earlier. Other than this, I ended up also having to convert some numeric data stored as a string back to numeric data. Other covariates are to the left.


```
Anemometer
Barometer
Dewpoint
Heat.Index
Hygrometer
Rain.Rate
Solar.Rad.Sens
Thermometer
UV.Rad.Sens
Wet.Bulb.Temp
Wind.Chill
Wind.Vane
Rain.Gauge
```

The next stage of data manipulation comes from converting the timestamps to a covariate. Essentially, I first had to parse the strings of dates into the Date type in R. Then, I had to determine how to approach converting Dates to a covariate. My initial thought was some method of correlating seasons with the dates so that the models can consider January 1, 2019, and December 31, 2017, similar, even though there exists a year difference between the two dates. Eventually, I settled on converting the dates to values through sine and cosine. First I converted the Dates to a numeric value, which is the number of seconds between the date and some other arbitrary date – the default is 1/1/1970, 12:00 AM from what the documentation says. After, I calculated the respective sine and cosine of these numeric values as $\sin\left(\frac{x*2\pi}{y}\right)$ and

$\cos(\frac{x*2\pi}{y})$, where x is the converted Date to numeric and y is the number of seconds within a year. As a result, dates half a year apart are considered furthest apart via these two covariates, a fair assumption that considers the seasons. In turn, the period of these functions is one year, as it should be to provide the correct seasonal values. As a result, both the sine and cosine data were added as covariates, bringing the total number of covariates to 14.

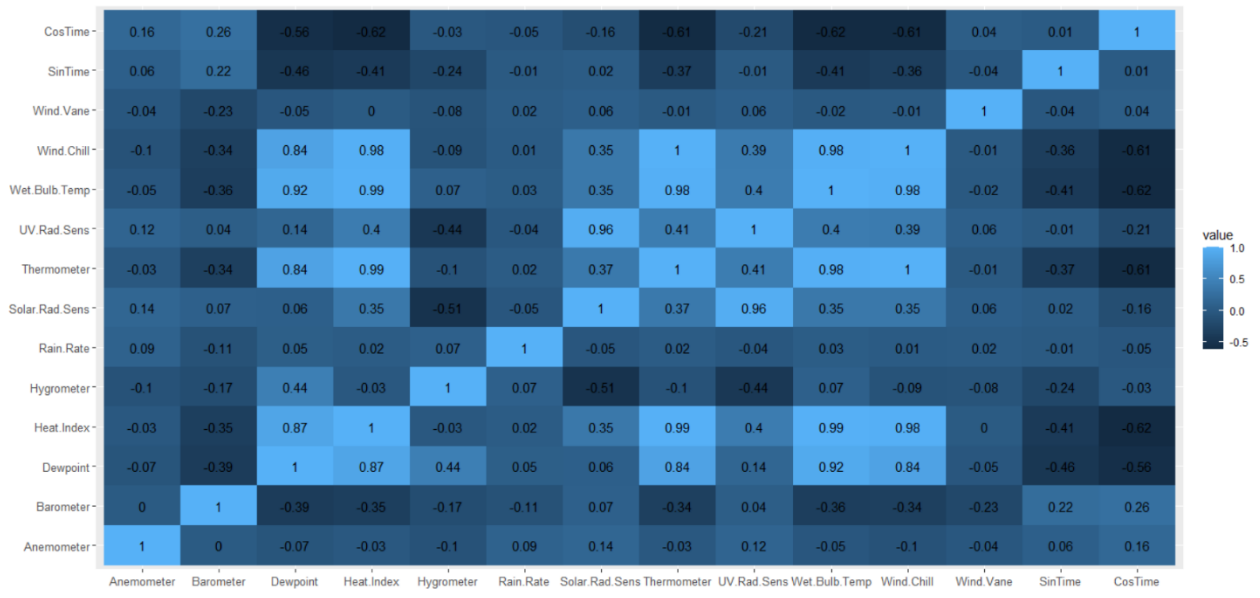
In addition, when cleaning the data, I noticed that some of the times are missing. After comparing a sequence of times from the beginning of the data to the end and comparing the times within the actual dataset, I noticed that nearly every day from mid-2017 till the present is missing ~8 hours of the potential 24 hours of data. Looking more into the missing dates, all these dates tend to be every third hour of every day. Luckily this is not a significant issue. Since we are not strictly looking at time-series methods for this project, we do not need to necessarily impute all these missing hours. In addition, in the following paragraph, I discuss the rearranging of the predictor variable for different hours. Since all the shifts shown below are divisible by 3, none of these missing times will affect the shifting below and there is no need to impute the missing values.

To construct the response variable for the 4 different times in the future (6 hours, 12 hours, 24 hours, 48 hours), I took multiple steps in storing the data. First, I stored the response variable (rainy conditions or no as a 1 or 0) within a hash map, with the key being the date as a string. We will make a copy of the full response list as y_xhrs, with x being the number of hours into the future specified. Then, for the length of the new list, if there exists a value in the hash map for the key of that index's day + x hours into the future, set that y_xhrs[i] equal to this value. Otherwise, add this index to an array of removed indices, which we remove once the for loop completes.

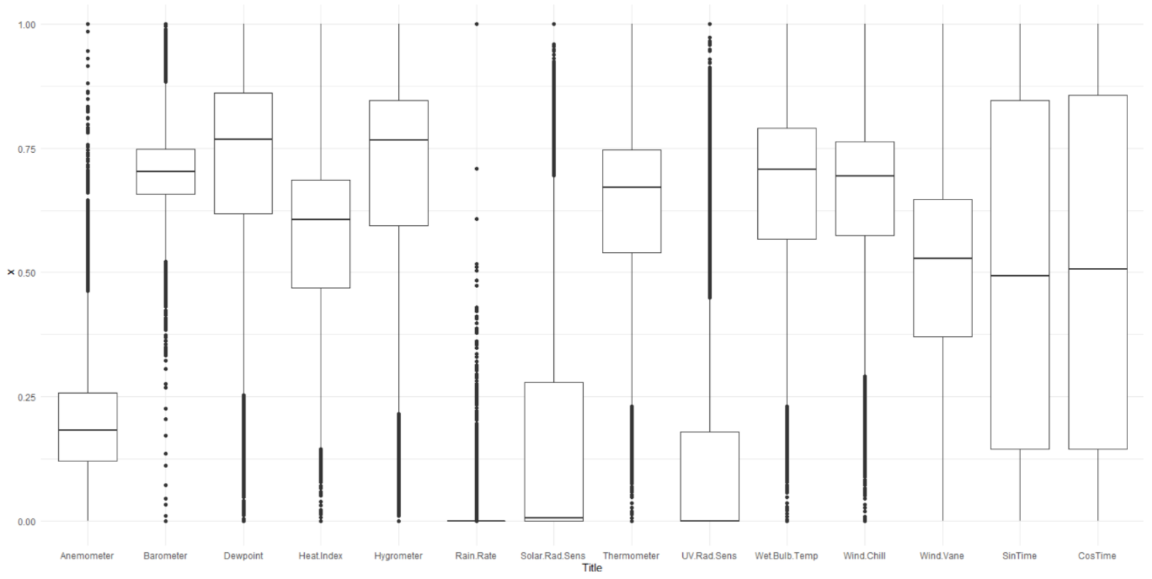
Lastly, I simply applied Min-Max Normalization to all the covariates. In turn, this brought the range of each covariate to become 0 to 1.

Exploration of Covariates:

After looking at a heatmap, there does appear to be a few highly correlated covariates. Nearly all have some reasonable explanation for their correlation (i.e. heat index and temperature having a high correlation). The correlation heatmap is below:

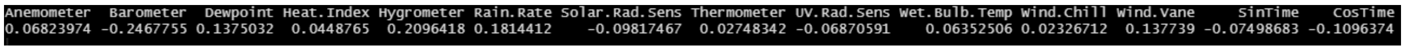


In addition, I also looked at a box-plot distribution of the covariates, shown below:



As shown here, there does appear to be some relatively interesting covariates. For instance, Rain.Rate appears to be nearly 0 most of the time, only being non-zero a slim amount of time. Previously, I did discuss the intricacies of this covariate and its relationship to Rain.Gauge, the variable the response variable is based off. Perhaps the models might see a non-zero Rain.Rate as a great predictor of future rainy conditions. In addition, the Soar.Rad.Sensor and UV.Rad.Sensor covariates both analyze some aspect of sunlight, hence how similar in distribution they are, as well as their low medians (half of any day is typically dark).

The one other exploratory measure utilized was viewing the correlation between the above covariates and the response variable (rainy conditions or no rainy conditions). Interestingly, the highest magnitude correlations came from the Barometer (at ~ -0.25), the Hygrometer, and Rain.Rate. Rain.Rate and the Hygrometer seem reasonable to be correlated to rainy conditions, but the Barometer does seem interesting, and it would be fascinating to see if this does play a factor in prediction. The correlations are found below:


```
Anemometer   Barometer   Dewpoint Heat.Index Hygrometer Rain.Rate Solar.Rad.Sens Thermometer UV.Rad.Sens Wet.Bulb.Temp Wind.Chill Wind.Vane      SinTime     CosTime
0.06823974 -0.2467755 0.1375032  0.0448765  0.2096418 0.1814412     -0.09817467  0.02748342 -0.06870591     0.06352506 0.02326712  0.137739 -0.07498683 -0.1096374
```

Models:

Before I discuss the models analyzed, I would like to first discuss metrics. Mainly, accuracy is certainly a large part of deciding the performance of specific models. However, I would also like to mention the false-negative rate (FNR). Going back to the initial question, I am considering success in this case to be rainy conditions (i.e. it might be wise to bring an umbrella). I might prefer a model that gives a small false-negative rate (i.e. will not predict a lack of rainy conditions when there exist rainy conditions in reality). As a result, I would like to analyze both accuracy and FNR and base a decision about a superior model on these two metrics. Below will analyze accuracy as error, as error = 1 – accuracy.

Regarding the classification frameworks, I decided to look at 9 different models:

1. A simple Logistic Regression model using glm()
    a. This is a relatively simple model, a good baseline to see if there exists a simple linear tendency between the covariates and the response variable.
2. An LDA Model, with all non-necessary parameters at default (i.e. has only the following variables set: lda(y~., data, family))
3. A QDA Model, with all non-necessary parameters at default (i.e. has only the following variables set: qda(y~., data, family))
    a. The above two models were selected to understand if any additional assumptions could provide a better model, for a linear and non-linear model, respectively.
    b. Note: For the above three models, since all are relatively quick to train and test, I decided to resample the training and testing data 50 times, averaging the error amongst them.
4. A KNN Model with K chosen through cross-validation for k between 1 and 25. I added this model in order to see if a non-linear localized approach might exceed expectations.
    a. All models did choose K=1
5. A simple decision tree, with tree being pruned at a specific height chosen via cross validation.
6. A Random Forest, with parameter mtry (number of covariates sampled as candidates) = 5, with 500 trees.
7. A GBM model, 5000 trees, interaction depth set to 1, shrinkage set to 0.001
    a. All 3 Tree-Based models were implemented as this data might perform very well for a binary response variable. In addition, I was interested to see how the RF model would compete with a GBM model.
    b. After looking initially at how the above 7 models performed on the 6-hour data, I decided to make 8 and 9. The goal of both ensemble models is mainly meant to produce a small FNR, even at the cost of an uptick in error. The justification for 8 comes mainly from KNN and an RF performing best for this dataset.
8. Ensemble.1: An ensemble model looking at the predictions made by the KNN and RF models. If either model said that there were rainy conditions x hours ahead, this model will agree and predict rainy conditions. We call this Ensemble.1.
9. Ensemble.2: An ensemble model looking at the predictions made by the 1st 7 models. If any model said there were rainy conditions x hours ahead, this model will agree and predict rainy conditions. We call this Ensemble.2.

All models above were used to predict 6 hours, 12 hours, 24 hours, and 48 hours into the future in separate iterations of the trial. All models then had their error and FNR calculated (for the 1st 3 models, the FNR is calculated using the last generated predicted values).

Regarding the choice of models, I will first discuss the reasons for excluding specific models. In particular, I decided not to include any SVMs, mainly due to the long training time for each SVM model. In turn, this would not be feasible to repeatedly run in case of debugging and re-running code to review different outputs. I also decided against running any PCA on the above data, as the number of rows/observations within the dataset compared to the number of covariates is astronomical, meaning no dimensionality reduction is especially needed in this scenario.

Model Analysis:
Below are the statistics for each model:

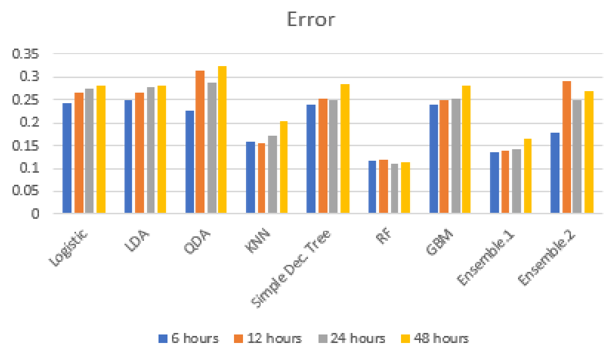
6 hours	Logistic	LDA	QDA	KNN	Simple Dec. Tree	RF	GBM	Ensemble.1	Ensemble.2	Base:
Error:	0.2425402	0.24822	0.22732	0.1578	0.2390977	0.1163	0.23991	0.13476	0.178022	0.2775
FNR:	0.7179903	0.73987	0.57739	0.2934	0.8213128	0.3128	0.82415	0.1705835	0.1405997	

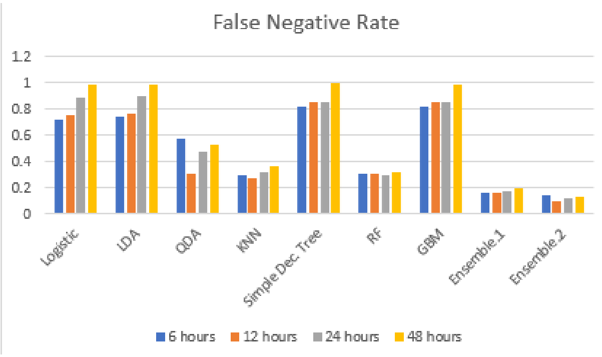
12 hours	Logistic	LDA	QDA	KNN	Simple Dec. Tree	RF	GBM	Ensemble.1	Ensemble.2	Base:
Error:	0.2666998	0.26653	0.31305	0.1554	0.2508679	0.1179	0.25006	0.1393196	0.2910206	0.28292
FNR:	0.75444	0.7594	0.30715	0.2741	0.8553121	0.3055	0.84828	0.1653576	0.0988012	

24 hours	Logistic	LDA	QDA	KNN	Simple Dec. Tree	RF	GBM	Ensemble.1	Ensemble.2	Base:
Error:	0.2755504	0.27862	0.28709	0.1697	0.2500289	0.1082	0.25153	0.1409885	0.2474823	0.29378
FNR:	0.8839323	0.89591	0.47542	0.3189	0.8492359	0.3015	0.85626	0.1763734	0.1243288	

48 hours	Logistic	LDA	QDA	KNN	Simple Dec. Tree	RF	GBM	Ensemble.1	Ensemble.2	Base:
Error:	0.2821468	0.28206	0.32395	0.2021	0.2853173	0.1144	0.28265	0.1636174	0.2699166	0.28404
FNR:	0.987013	0.98336	0.53612	0.3612	1	0.3259	0.98701	0.2000812	0.1278409	

Please note that the Base column indicates the accuracy given we say there are not rainy conditions on any given day. As a result, all the models (except QDA for 12, 24, and 48 hours) are more accurate than systematic guessing.
For a comparison of the various models visually, please see these charts:





Conclusions:

As shown above in the various charts and figures, the KNN and RF models outperformed the other non-ensemble margins by a wide margin, indicating non-parametric methods seem to outperform the other various models. Interestingly, the RF model consistently had the lowest error out of all the models. However, the Ensemble.1 model (combining the KNN and RF predictions), while having a ~3-5% higher error, produces a significantly lower FNR than the FNR of the next smallest model (at least 1/3 less than the FNR of the RF model consistently), perhaps indicating a superior model.

The models do tend to have a higher error for farther predictions, which was expected. However, the magnitude of this higher error is very slight. For most models, this uptick is only ~3-4% higher between predicting 6 hours in advance and 48 hours in advance. Nonetheless, at 48 hours in advance, this small uptick raises most of the models' error (excluding KNN, RF, and the ensembles) to be at the base level, indicating that these models are essentially ineffective at producing a lower error than random guessing. In addition, the FNR's of these models at this higher end are nearly 1 (except QDR, which maxes at 0.5), indicating once again similarity to the systematic guessing model. The exception to this uptick in error rules comes with the RF model, which was able to hold a consistent 11% error. In addition, I found incredibly surprising the GBM model having an incredibly poor

So, in sum, the two most successful models were the RF and Ensemble.1 models, with RF being better just for accuracy, and Ensemble.1 being the best for a combination of accuracy and FNR. In addition, as the prediction windows increased, the accuracy and FNR of the models (except KNN and Ensemble.2) approached the base systematic guessing accuracy and 1.0, respectively. As a result, other than the 4 models that were able to predict for long

Future Work:

There does exist a few tempting future developments if work on this subject is to continue. Mainly, these consist of adding different models or techniques.

- First and foremost, a Recurrent Neural Network model (RNN) is a framework that is a fantastic ultra-complex model to make long-term predictions on time-series data. If I were to repeat this investigation, it would be necessary to impute the missing datapoints within the UF dataset.
- Second, given the ability of the RF model to produce such a fantastic model for all prediction windows, I would be interested in performing cross-validation on the RF model and see if this produces a difference in error and FNR.
- Third, I could have added another covariate: whether there exists rainy conditions at the present time (as opposed to x hours in the future).

Other than these future tweaks & inclusions listed, I believe that the investigation was thorough enough for the data provided. If more accurate and documented data of the same caliber is found, a more thorough investigation could more likely be undertaken.