STAT 190 – Case Studies in Data Analytics

TO: PROFESSOR LENDIE FOLLETT
FROM: CHRIS CAVAN, MARSHALL NDHLOVU, NATHAN GOTTWALS, NOAH MADDIO
SUBJECT: PROJECT 1: PREDICTING FAILURES OF BHE WIND TURBINES
DATE: 5/10/2023

1. Introduction

The purpose of this analysis was to investigate the failure of wind turbines for Berkshire Hathaway Energy (BHE). The datasets we received from BHE concerned a group of their wind turbines located here in Iowa and can be broken down into three types. Note that we have elected to work with the **second** batch of data received on April 3, 2023 for our analysis.

First, the work order dataset contained logs from technicians and mechanics that were physically at the turbines conducting maintenance or giving repairs. These records told us when each turbine required resources from BHE including labor, expensive parts, and lost potential energy by stopping the turbines and sending a team out to them. This singular dataset consisted of 75 observations and 8 columns.

Next, the fault code time series provided logs from the computers and sensors within the turbines of when a warning or error occurred, specifying the exact time and reason for the fault. These records help detect what specific components are failing in the turbines and the severity of it. The fault code time series was divided into numerous files, which had to be aggregated together to have over 700,000 observations and 7 columns.

The rest of the data includes sensor readings from each of these components within the turbine, including temperatures, pressure, and power output. These datasets, which also record the times of the readings, can be matched up with the fault code time series to show exactly what was happening in the turbines at the time of these errors. These datasets also were divided into numerous files for each reading, with a highly variable number of observations for each reading.

This paper aims to discover relationships between the various sensor data described above and the existence of critical faults to understand the causes of wind turbine failures.

The following topics will be addressed in this whitepaper:

1. Problem Description
2. Model Variable Definitions and Variable Explorations
3. Model Specification and Justification
4. Results- Interpretation of Model Output and Variable Importance

1.1 Business Problem

The main business problem we are looking to address for BHE is ultimately to save money and resources, as well as maximize their potential power output. By analyzing why their turbines might have stopped or broken in the past, we look to find a reliable way to limit the need for maintenance or repairs on them. The longer the turbines can run without sustaining any damage or other issues, the more efficient they will be for BHE in producing clean, sustainable energy for their customers. The goal of this analysis and report will be to help protect their wind turbines from dangerous conditions that could lead to additional work orders and expensive maintenance for the company.

1.2 Analytical Problem

The analytical problem is to build a predictive model that best fits both a training and testing subset of this turbine data. Given a set of turbine sensor readings, the model must effectively estimate when a critical fault is likely to occur. We aim to solve this problem by building and tuning a random forest model on a modified version of the original data set where all sensor readings have been lagged back by 6 hours. This will allow us to model the conditions leading up to critical faults rather than just predicting the critical faults in real-time, or after they occur.

2. Methods - Exploratory/Validation

2.1 Data Aggregation

Our first objective was to clean the data and get it ready for exploration. This involved aggregating all sensor reading datasets together and joining them with the fault code time series. For each of the sensor datasets, data was read into their own data frame. The DateTime column was then converted to a date format with a Tidyverse function. Because of the large size of the data, we looked to condense observations into time intervals of 1-hour. Using the DateTime column, we rounded observations into their nearest 1-hour interval with the lubridate package, creating a new column for our timestamps named round_date.

Work orders were cleaned by removing observations with duplicates while keeping the orders that had component_type filled out. The fault code dataset was cleaned to keep only the first occurrence of fault codes and then Time_Diff, DateTime, Date, StatusCode columns were dropped from the data frame. This way we didn't take it into consideration if a type of fault was pinged numerous times in a short-span for the same initial error.

For the sensor data, the summarise() function was used to create new columns for the mean, max, min, and standard deviations of all the sensor readings within each 1-hour interval, and

duplicates were removed. Because we had condensed multiple observations into a single reading for each hour, extracting the min, max, and standard deviations of each interval ensured that we still included any extreme values or high variability that may have occurred within that time period. The sensor data, except for wind speed readings, which we removed due to 78% of its observations holding missing values, was then joined together by a full join by its location_id and round_date to a dataset name fully_joined. The fault code dataset was then left joined to this data. Our final result was a dataset of the summary statistics for each sensor reading by hour, including whether or not a fault occurred during that hour and its description.

2.2 Data Exploration

After aggregating and exporting our joined sensor reading and fault code dataset, we were ready to look at how our explanatory variables interact with each other and the presence of a critical fault. After playing around with different boxplots and scatterplots to get an idea of the overall distributions of the data, two stuck out to us that we found particularly useful:

Figure 1: Standard Deviation of Generator RPM Vs Hydraulic Pressure by Critical Faults

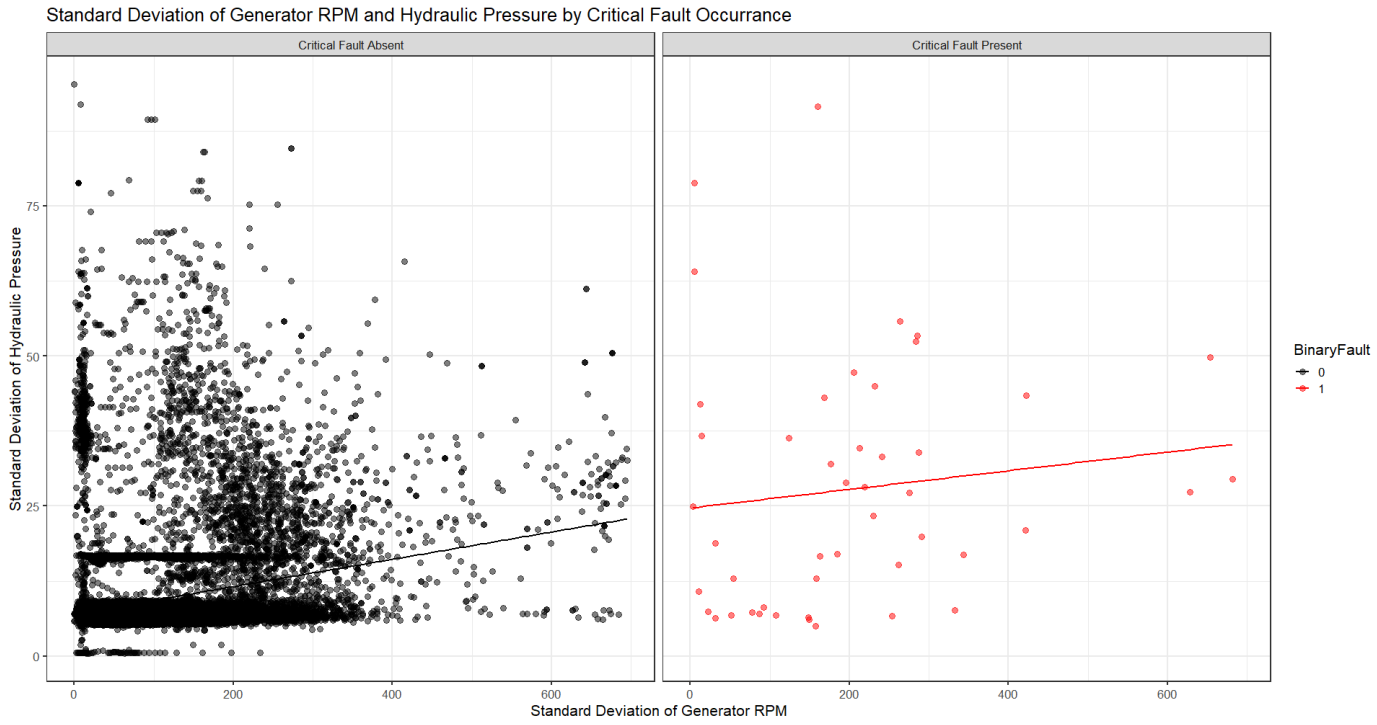


Figure 1 above shows the standard deviation of hydraulic pressure and generator rpm plotted against each other and split into observations with and without a critical fault. From the density of transparent points in the 'Critical Fault Absent' graph, we observed that the majority of observations fall into a trend of low standard deviation for hydraulic pressure. However, looking

at the ‘Critical Fault Present’ graph on the right, we saw that the majority observations resulting in critical faults displayed higher values for standard deviation of hydraulic pressure. This is confirmed by the trend lines of each graph, for which the red line starts and ends at noticeably higher values than the black line. This provided evidence that 1-hour intervals which experienced more variation in its hydraulic pressure were more likely to result in critical faults and could prove useful for prediction in our model.

Figure 2: Minimum of Active Power Vs Hydraulic Pressure by Critical Faults

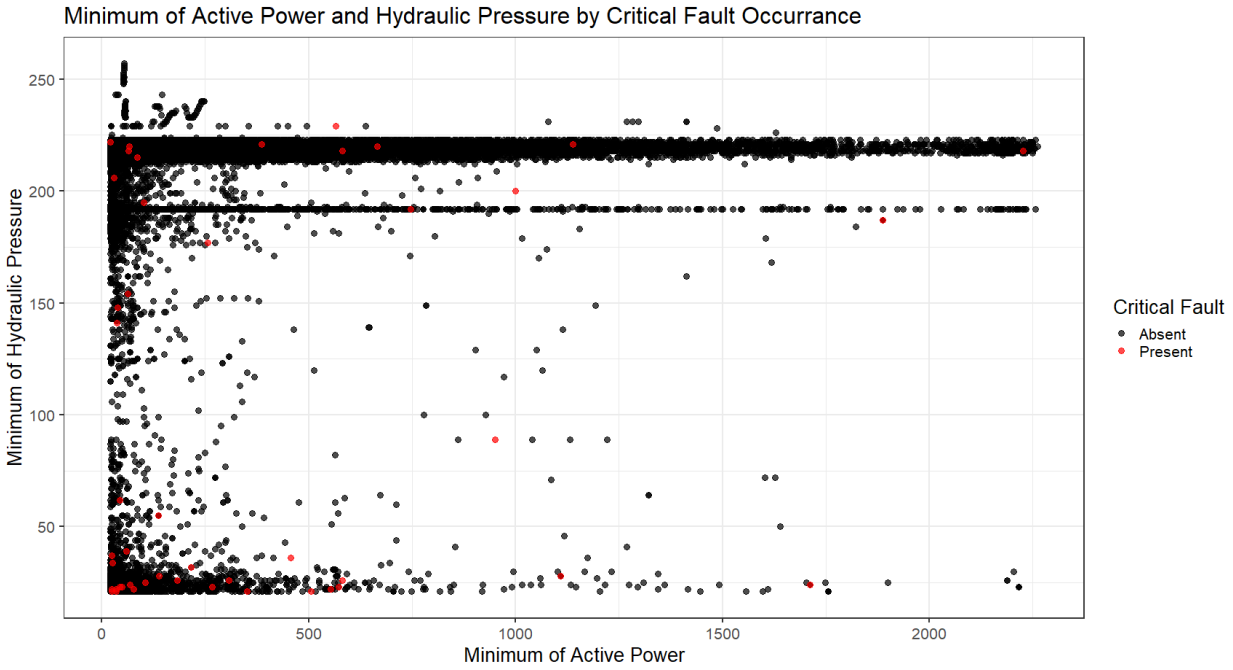


Figure 2 above shows the minimum value of hydraulic pressure and active power plotted against each other and colored by the presence of a critical fault. Minimum value refers to the lowest recorded observation within each summarized 1-hour interval. From this scatterplot, we saw that the majority of critical faults happened when there were low, almost zero, minimum values for hydraulic pressure and nearly all critical faults occurred when the minimum active power was low/under ~500. This was something we didn’t really expect at the start of our exploration, as generally one might think that unusually high pressure or power output readings would lead to these critical faults. However, this showed us that it is actually of more concern when these turbines are producing unusually low readings, meaning that including the minimum values in our model could help it to predict critical faults better.

2.3 Missing Data

As a whole, the dataset had very few observations that weren't missing at least one sensor reading at any given point in time. Therefore, in order to build a model of the data, we looked to impute accurate values for these readings into the missing gaps. As noted in the joining section, the wind speed data was not included in our aggregated data set as it had too few actual observations from which the imputing function could build upon. Then, from the imputeTS package, we filled in our missing values using kalman smoothing, which utilizes past observations and the values around the NA's to more accurately impute the missing data.

2.4 Lagged Variables

Another step we took in our data manipulation to provide more value in our results was to lag the values for our sensor readings back by 6 hours (or 6 rows/observations with our 1-hour intervals). The purpose of this idea was that it's not particularly useful to predict whether a critical fault is happening right now because at that point it could be too late. If the problem is already occurring within the turbine it can cause stops or potential damage. By looking at the association between critical faults occurring and the corresponding sensor readings from 6 hours earlier, we can predict what specific conditions within the turbine resulted in errors just hours later and prevent other turbines from reaching them in the future. We achieved this with the lag() function from the dyplr package, first grouping by turbines to ensure observations from one turbine did not accidentally push onto another. The rows that now contained missing values from shifting the data were removed from the dataset.

2.5 Response Variable

A new column named FaultCodeType was created to categorize the fault codes into 3 different categories. The critical category is fault codes that were identified by BHE as expensive faults that would require a stop and work would likely need to be done on the turbine. The second category would be warning, these are fault codes that mostly indicate warnings from the turbine or were related to the more crucial parts, meaning a critical fault code may occur if left unchecked. The last category for is other, this is for when no fault code exists or fault codes that occur when there is a stop or identification for insignificant occurrences such as untwisting.

From the FaultCodeType (which was utilized mostly in exploration), we created a binary variable, named BinaryFault, which simply indicated 0 for a warning or other fault code, and 1 for a critical fault code. Because our sensor readings were lagged back 6 hours, we were already analyzing the conditions leading up to critical failures and therefore found little significance in

trying to predict the warning faults as well. The new BinaryFault variable became the focus/response of our predictive modeling.

2.6 Explanatory Variables

There will be 33 numerical explanatory variables in the final data set used to predict our dependent variable BinaryFault. The first explanatory variable is location_id, which records the specific turbine from which the sensor readings are pulled. No changes were made to this variable, and it was included because we wanted to see if there were significant differences when predicting critical faults between individual turbines. We also used all nine of the original sensor reading types besides wind speed, which was removed in its entirety due to data incompletion. Each sensor reading data set was aggregated at 1 hour intervals, imputed, and lagged 6 hours as described above. For each sensor reading type, the mean, maximum, minimum, and standard deviations for the previous time windows were included as explanatory variables. A description of each sensor reading type can be found in the table below.

Sensor Reading Type	**Variable Names in Model**	**Description**
Gearbox HS Bearing Temperature	lag_XXX[1]_bearing	Temperature of the high-speed gearbox bearing measured in degrees Celsius
Hydraulic Pressure	lag_XXX_hydraulic	Hydraulic pressure of the wind turbine generator measured in bar (a metric unit of pressure)
Generator RPM	lag_XXX_rpm	Number of revolutions completed per minute of the wind turbine generator
Gearbox Oil Temperature	lag_XXX_oil	Temperature of the gearbox oil measured in degrees Celsius
Gearbox IMS Bearing 1 Temperature	lag_XXX_ims	Temperature of the first intermediate shaft bearing measured in degrees Celsius
Gearbox IMS Bearing 2 Temperature	lag_XXX_ims2	Temperature of the second intermediate shaft bearing measured in degrees Celsius
Active Power	lag_XXX_active	Amount of power produced by the wind turbine measured in kW (kilowatts)
Ambient Temperature	lag_XXX_ambient	Environmental air temperature measured in degrees Celsius

[1] XXX in the variable names above is replaced with mean, max, min, and sd in the final data set

2.7 Final Data Set

The final aggregated data set was used to train and test our random forest model and contained 29,443 observations with 34 total variables. Each row contains our binary Y variable indicating critical faults, our location_id variable to indicate the location of the sensor readings, and numerical sensor data for the 1-hour period that is 6 hours before the indicated fault. The numerical sensor data consists of the mean, maximum, minimum, and standard deviations for each of the 8 sensor readings described in section 2.5. The raw sensor data values were not present in the final data set used in our random forest model.

2.8 Other Methods Explored

2.8.1 Fuzzy Matching on Work Order Data

To implement fuzzy string matching, the use of the reticulate package was needed in order to use the python method, difflib. The difference between two strings, between a work order information and a list of fault code description, is calculated and compared between scores of the fault code description and then the best scored fault code description gets paired to the work order information. While this worked relatively well, we came to the conclusion that incorporating the work order data into the fault code time series would not benefit the type of predictive model we were looking to build. Rather than try and predict when a work order was going to happen as a result of a specific fault, we were more concerned with predicting the presence of the group of faults that were deemed by BHE to be critical (lead to costly maintenance and repairs).

2.8.2 Anomaly Detection Models

One of the other methods we considered to help us explain what conditions were leading to critical faults was an anomaly detection model. The goal here was to see if there were extreme values for our sensor readings that would be deemed outliers and were causing problems within the turbine. Anomaly detection models can provide visuals of these events and breakdown trends in the data. We used the ‘anomalize’ package to start this process but quickly ran into concerns about its utility.

The process requires data to be given at the day to day level. As our data was broken down into 1-hour intervals, we found that we would lose a lot of our data quality due to how much readings can vary within a 24-hour period. We were more interested in breaking down the behavior of the turbines throughout each day, and therefore did not find value in an anomaly detection model for our purposes. Additionally, the model can only handle so many observations before it becomes computationally and time intensive, meaning it is only really feasible to look at small windows in

time. Since the critical faults we were looking to model are rare incidents, we were more interested in a model that is able to incorporate all of the given data.

3. Methods - Predictive Modeling

Two distinct random forest models were considered as candidates for this task. The first candidate model does not include location_id as an explanatory variable, whereas the second candidate model does. Both models used a random 80/20 split to divide the data set into the training and testing sets, respectively. A seed value of 231231 was used for both models when randomizing the order of the data prior to dividing the data set. After training and testing both models, the final random forest model was selected as the model without location_id. This is because the original scope of the project was to establish one predictive model for all turbines, as opposed to creating individual models for each turbine. However, we were still interested to see how the inclusion of location_id would affect the random forest output, even though model 2 is not as useful as model 1 after knowing the context of this problem.

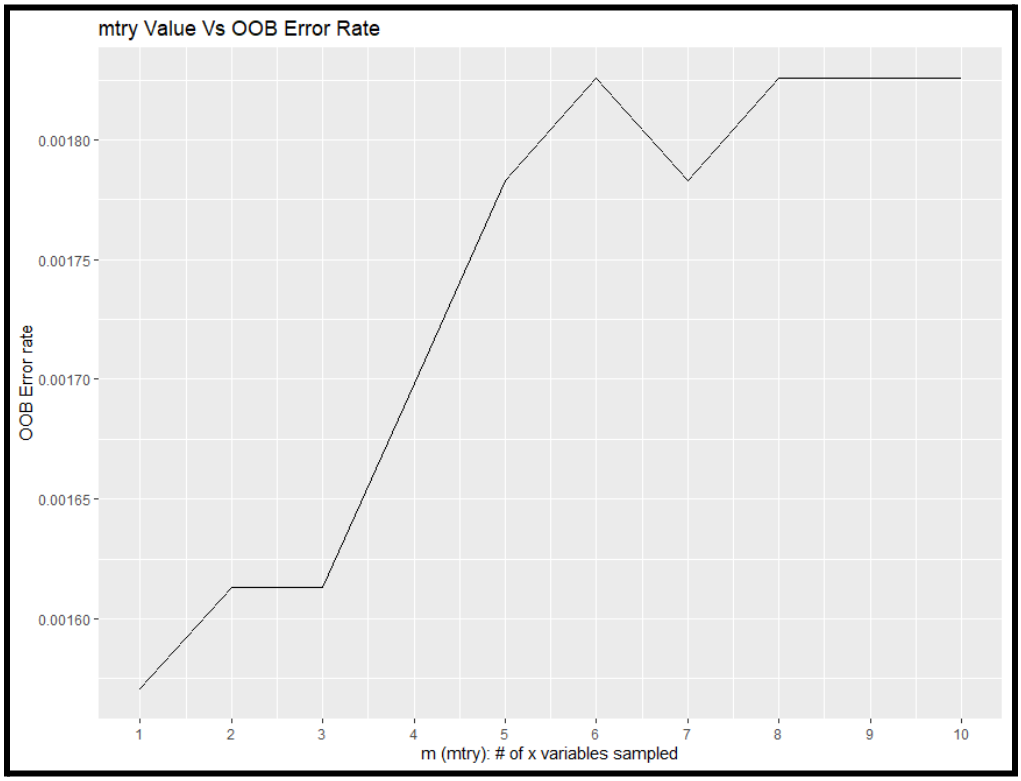
3.1 Model 1 Specification

The first random forest model utilized in this analysis had the 32 numeric sensor reading variables without location_id. These variables as named in the model can be seen below:

X Variables:		**Y Variable: BinaryFault**	
lag_mean_bearing	lag_max_bearing	lag_min_bearing	lag_sd_bearing
lag_mean_hydraulic	lag_max_hydraulic	lag_min_hydraulic	lag_sd_hydraulic
lag_mean_rpm	lag_max_rpm	lag_min_rpm	lag_sd_rpm
lag_mean_oil	lag_max_oil	lag_min_oil	lag_sd_oil
lag_mean_ims	lag_max_ims	lag_min_ims	lag_sd_ims
lag_mean_ims2	lag_max_ims2	lag_min_ims2	lag_sd_ims2
lag_mean_active	lag_max_active	lag_min_active	lag_sd_active
lag_mean_ambient	lag_max_ambient	lag_min_ambient	lag_sd_ambient

Model 1 had an ntree[1] value of 1000 and an mtry value of 3. In other words, our random forest model was composed of 1000 individual decision trees and 3 of the 32 available explanatory variables were considered at each split in the individual trees. We chose our ntree value of 1000 to reflect accepted industry practices for random forest modeling. Our mtry value was tuned by trying all possible values from 1 to 10 and recording the corresponding Out-of-bag (OOB) error rate. Commonly, an mtry value equal to the square root of the number of explanatory variables in the model is used. Since we have 32 explanatory variables, this would indicate an mtry value of between 5 and 6. Thus, we wanted to test values adjacent to 5 and 6 so the range 1 to 10 was chosen. The results of this tuning procedure can be seen in figure 3 below.

Figure 3: Tuning mtry for Random Forest Model 1



We wanted to choose the mtry value that minimized the OOB error rate. Interestingly enough, an mtry value of 1 resulted in the lowest OOB error rate of 0.001570858. However, this would mean that the split variable at each split in the decision trees is completely random which could lead to potentially biased results that we wanted to avoid. Thus, we selected our mtry value of 3 which had the next lowest OOB error rate of 0.001613314.

[1] Note that B is commonly used to represent the number of trees in statistics. We will use ntree here.

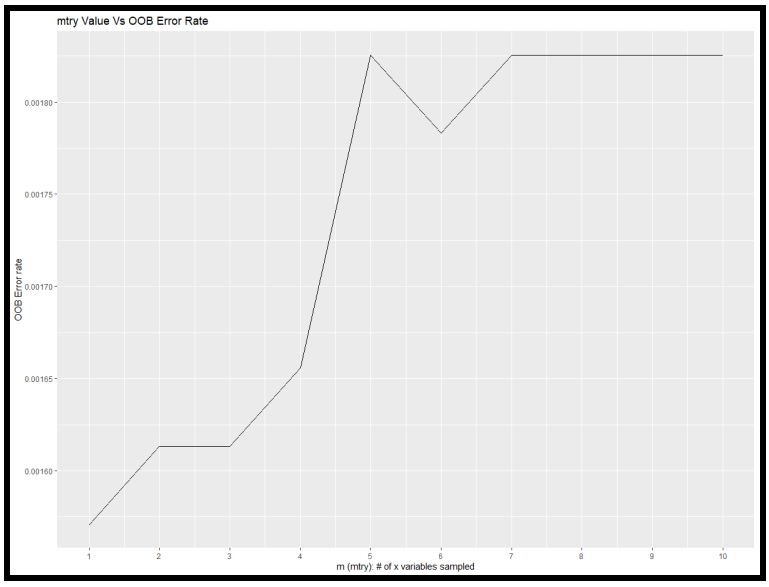
3.2 Model 2 Specification

The second random forest model considered in this analysis had the 32 numeric sensor reading variables included with location_id as explanatory variables. These variables as named in the model can be seen below:

X Variables:	*location_id*	**Y Variable: BinaryFault**	
lag_mean_bearing	lag_max_bearing	lag_min_bearing	lag_sd_bearing
lag_mean_hydraulic	lag_max_hydraulic	lag_min_hydraulic	lag_sd_hydraulic
lag_mean_rpm	lag_max_rpm	lag_min_rpm	lag_sd_rpm
lag_mean_oil	lag_max_oil	lag_min_oil	lag_sd_oil
lag_mean_ims	lag_max_ims	lag_min_ims	lag_sd_ims
lag_mean_ims2	lag_max_ims2	lag_min_ims2	lag_sd_ims2
lag_mean_active	lag_max_active	lag_min_active	lag_sd_active
lag_mean_ambient	lag_max_ambient	lag_min_ambient	lag_sd_ambient

Model 2 had an ntree value of 1000 and an mtry value of 3. We once again chose our ntree value of 1000 to reflect accepted industry practices for random forest modeling. Our mtry value was also tuned by trying all possible values from 1 to 10 and recording the corresponding Out-of-bag (OOB) error rate. The results of this tuning procedure can be seen in figure 4 below.

Figure 4: Tuning mtry for Random Forest Model 2



Once again, we wanted to choose the mtry value that minimized the OOB error rate, and once again, an mtry value of 1 resulted in the lowest OOB error rate of 0.001570858. Finally, we selected our mtry value of 3 which had the next lowest OOB error rate of 0.001613314.

3.3 Model Justification

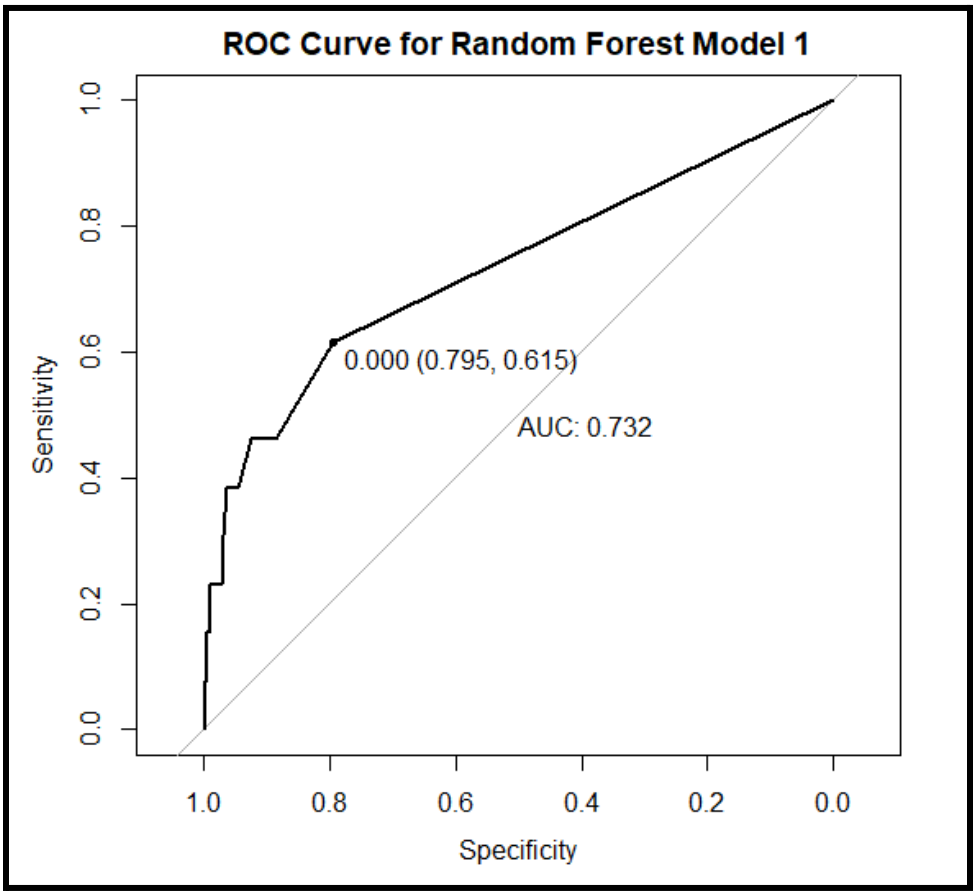
The proposed random forest models are appropriate for our final aggregated data set for a number of reasons:

1. Random forest models predict output with high accuracy for large datasets of high dimensionality, such as our dataset, which has 29,443 observations and 33 variables.
2. If relationships between our explanatory variables are non-linear, traditional linear regression models will not capture the complexity of the data. In the data, one example of non-linearity is the relationship between Generator RPM and Active Power. Here we saw a clear curve in the trend between the two. Random forests do not make any assumptions about the distribution of the data and therefore can be applied easily to model these relationships.
3. Random forest models can be readily applied to binary classification problems.
4. Random forest models combine the results of multiple decision trees, making it less prone to overfitting and much more accurate than a single decision tree. Overfitting was definitely a concern while building our model as we are using 32 explanatory variables, and the main purpose of our model is to be effective in predicted faults for any turbine, not just the ones we were given data for.
5. Its use of decision trees makes random forests much less sensitive to outliers, providing accurate predictions in the presence of extreme values. From our initial exploration of the sensor reading data, we observed some outliers and unusual points that would be handled well by a random forest.

4. Results

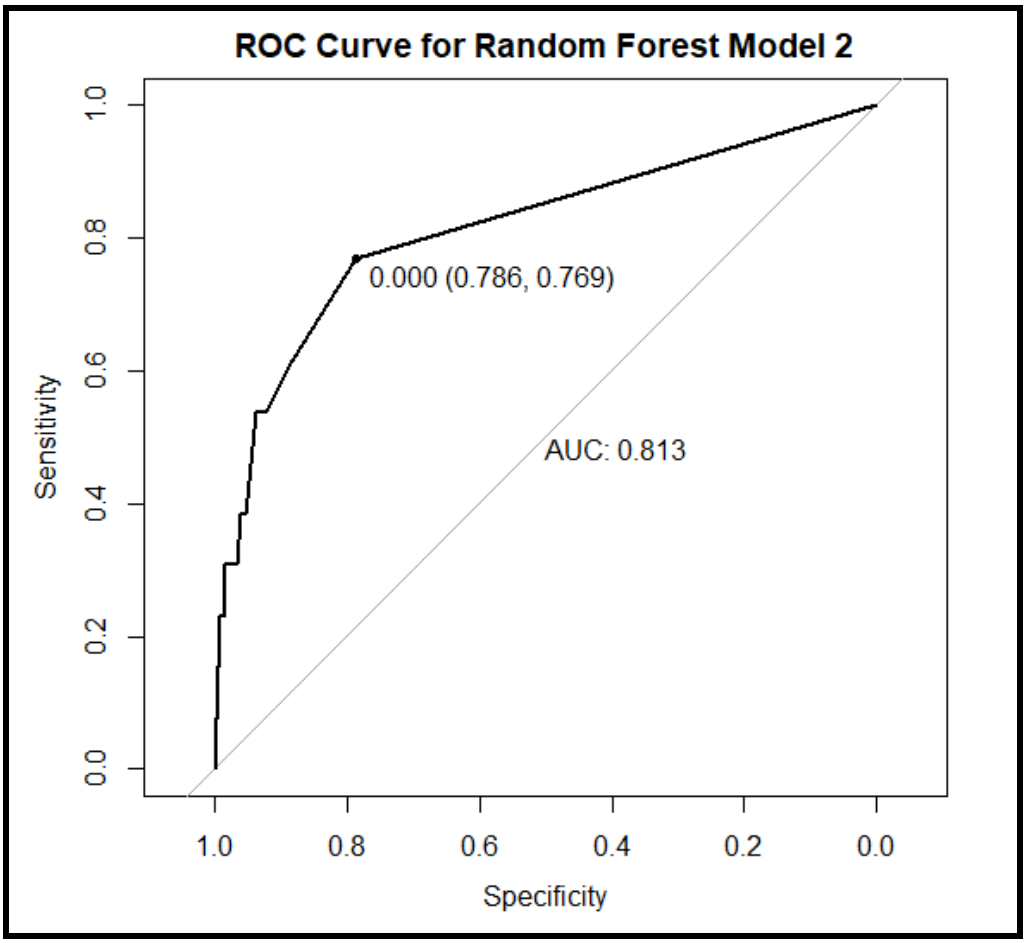
4.1 Interpretation of Random Forest Output

Figure 5: ROC Curve Output for Random Forest Model 1



From our ROC curve output for model 1 in figure 5, we can see that model 1 achieved a specificity of 0.795 and sensitivity of 0.615, with an area under the curve of 0.732. Furthermore, our pi star value of 0.000 can be interpreted as: If the random forest model predicts a critical fault with probability greater than 0.0001 (pi star) then we predict a critical fault. Otherwise, we do not predict a critical fault. This probability may seem low, but we know that critical faults are an extremely rare event occurring in our data set. Thus, our low pi star value makes sense. The specificity, or true positive rate, shows us that the model correctly predicts there is a critical fault when there really is one 79.5% of the time. The sensitivity, or true negative rate, shows us that the model correctly predicts the absence of a critical fault when there is not one 61.5% of the time. The AUC represents that there is a 73.2% chance that our model will be able to distinguish between the presence and absence of a critical fault.

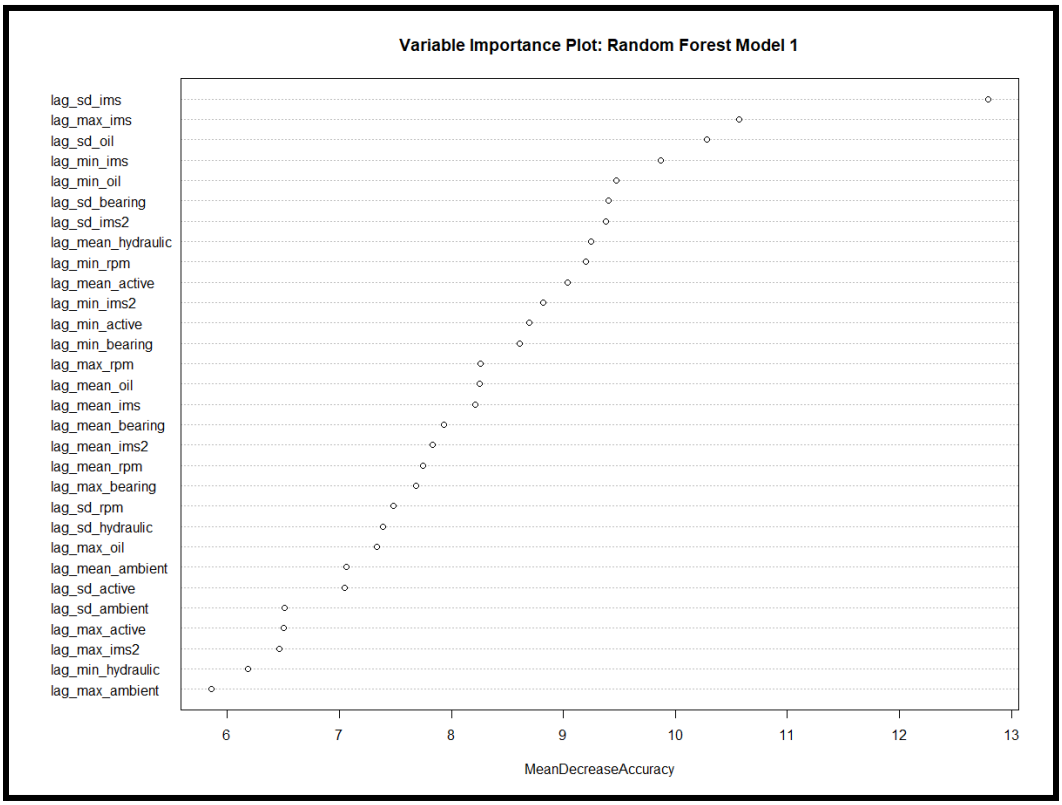
Figure 6: ROC Curve Output for Random Forest Model 2



From our ROC curve output from model 2 in figure 6, we can see that model 2 achieved a specificity of 0.786 and sensitivity of 0.769, with an area under the curve of 0.813. Furthermore, our pi star value of 0.000 can be interpreted as: If the random forest model predicts a critical fault probability greater than 0.0001 (pi star) then we predict a critical fault. Otherwise, we do not predict a critical fault. The specificity, or true positive rate, shows us that the model correctly predicts a critical fault when there really is one 78.6% of the time. The sensitivity, or true negative rate, shows us that the model correctly predicts the absence of a critical fault when there is not one 76.9% of the time. The AUC represents that there is a 81.3% chance that our model will be able to distinguish between the presence and absence of a critical fault.

4.2 Variable Importance for Final Model

Figure 7: Variable Importance Plot for Random Forest Model 1



The variable importance plot in figure 7 provides a list of the most significant explanatory variables included in our random forest model in descending order by a mean decrease in accuracy.

We can see that our 5 most important predictors of critical fault events are lag_sd_ims, lag_max_ims, lag_sd_oil, lag_min_ims, and lag_min_oil. It is interesting that 3 of the related readings for the temperature of the first gearbox IMS bearing are featured on this list, which implies that this sensor reading may hold the most predictive power overall. Another notable point is that the variables associated with the active power sensor data were not important predictors of critical fault events. We would have intuitively expected outliers in the power output data to be significant in predicting critical fault events. This variable importance plot reinforces the notion that not all statistical relationships are intuitive or obvious before running the random forest model.

5. Conclusion

In this project, we created a random forest model to predict critical faults within wind turbines using aggregated turbine sensor data from a 1-hour period that was 6 hours before the faults occurred. From our model results, we concluded that the temperature of the first Gearbox IMS Bearing was the most significant predictor of critical fault occurrence in the surveyed subset of Berkshire Hathaway Energy's wind turbines. However, the random forest models presented in this paper have notable limitations, and further model refinements or the inclusion of more explanatory variables would be required before this model can be put into production.

While the conditions for what constitutes an acceptable AUC vary greatly depending on the model and its implications, in the case of predicting critical failures on any given turbine, our model is not quite there yet. This model will still prove somewhat effective if implemented in a real world situation as our AUC of 0.732 means our model is certainly capable of discerning whether a critical fault is present or not. However, with an around 27% chance of the model failing to make that distinction and an actual true positive rate of 62%, the risk of missing a critical fault would likely be too great for BHE's purposes. With the cost of inaccuracy being expensive repairs, labor, and lost energy, a model with higher reliability would be preferred.

Once again, we were interested in the future potential of building separate random forest models for individual turbines. Although this fell outside of our original project scope, our initial analysis indicates that individual turbine models may be promising. This is because the AUC of our random forest model 2 with location_id included was 0.813, which is higher than the AUC of our random forest model 1 without location_id at 0.732. Furthermore, an analysis of the variable importance plot for model 2 (not pictured here) revealed that location_id was the most important variable in that random forest model. This is an interesting conclusion that would need to be explored in more depth outside of this course.