


ReNeWIND: WIND ENERGY PRODUCTION

**Wind Turbine Data Collection using Machine Learning
Techniques**

Model Tuning: Module 6

April 2024

AGENDA

- ❑ Executive Summary
 - ❑ Objective & Solution Approach
 - ❑ Data Details - Description
 - ❑ Data Overview - EDA & Data Preprocessing
 - ❑ Summary of Model Performance & Hyper Parameter Tuning
 - ❑ Model Building with Pipeline
 - ❑ Production Model
 - ❑ Insights & Conclusions
- 
- A series of three parallel white diagonal lines are positioned in the bottom right corner of the slide, extending from the middle of the right edge towards the bottom left.

Executive Summary

Reducing the impact of energy production is our primary focus at ReneWind. Wind energy is one of the technologies worldwide that has a concentrated force of technology.

The U.S. Department of Energy offers a guide to achieving Operational Efficiency using Predictive Maintenance Practices. There is no cost comparison between Replacement Costs and Repair Costs. Predictive Maintenance; the technology we use at ReneWind uses sensor information and analysis to measure and predict the degrading of parts and component capability.

Our ability to predict failure patterns based on the accurate prediction of component failure before it happens, is the most cost effective method. This means the cost of operation and turbine maintenance will be significantly lower.

Objectives & Approach:

Use Classification Models to Predict Failure/Breaking:

1. Build and Tune various models - find the best one that helps identify failures
2. The best model will predict/identify failures to reduce overall maintenance costs

ReneWind Data Scientists specifically using Machine Learning Techniques were tasked with working to improve the process of production on Wind Energy Turbines. We have collected generator failure data using sensors inside the turbines.

Encrypted Data was gathered from different companies using sensors from inside the wind turbine.

Data & Details:

- A straightforward approach was used within the Dataset, only the Numeric Sensor Variables

Numeric Sensor Related to Environmental Factors:

Wind Speed - Humidity - Temperature

- Dataset has 40 Predictors: Labeled V: Numbers1 - 40
- The Classification Model will make predictions (translation below):
 - True Positives (TP) - Correct Model Predictions {Repair Costs}
 - False Negatives(FN) - Actual Failures with no model Prediction {Replacement Costs}
 - False Positives(FP) - Detected Failure - but did not fail {Inspection Costs}
- The cost of a Generator Repair is much less than Replacing it. Also, an Inspection of the equipment is less than the cost of a Repair.

Data Description

- ❖ Data Provided was collected from Sensors inside engine of Wind Turbines. Different companies provided this data and it is a transformed version of the original
- ❖ Train Data - Used for Training and Tuning of Models
- ❖ Test Data - Only for Testing the Performance of the Final Best Model
- ❖ Both of these Datasets consist of 40 Predictor Variables and 1 Target Variable
- ❖ Train Set 20,000 Rows & Test Set 5,000 Rows
- ❖ TARGET VARIABLES
 - 0 = No Failure
 - 1 = Failure
- ❖ Negative numbers are values obtained from the sensors and are not abnormal values. These values will not be treated/imputed/dropped from our data

Data Overview

A sample of the Data and what the head of the dataset looked like when we received it for analysis. As mentioned previously we did use the negative values. These were actual numbers produced by the sensors.

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15
0	-4.465	-4.679	3.102	0.506	-0.221	-2.033	-2.911	0.051	-1.522	3.762	-5.715	0.736	0.981	1.418	-3.376
1	3.366	3.653	0.910	-1.368	0.332	2.359	0.733	-4.332	0.566	-0.101	1.914	-0.951	-1.255	-2.707	0.193
2	-3.832	-5.824	0.634	-2.419	-1.774	1.017	-2.099	-3.173	-2.082	5.393	-0.771	1.107	1.144	0.943	-3.164
3	1.618	1.888	7.046	-1.147	0.083	-1.530	0.207	-2.494	0.345	2.119	-3.053	0.460	2.705	-0.636	-0.454
4	-0.111	3.872	-3.758	-2.983	3.793	0.545	0.205	4.849	-1.855	-6.220	1.998	4.724	0.709	-1.989	-2.633

Data Overview

- The dataset contained only numeric data types in all Rows - Floats
- No Duplicate Values were within the data
- Missing Values:
 - Train Data had missing values in only two columns (V1 = 18 & V2 = 18)
 - Test Data had missing values in the same two columns (V1 = 5 & V2 = 6)
- Statistical Summary of the Dataset showed a wide range between negative and positive between Mean and Max.
 - V3 & V35 had the highest positive values, which says we have Outliers in the data

Data Overview

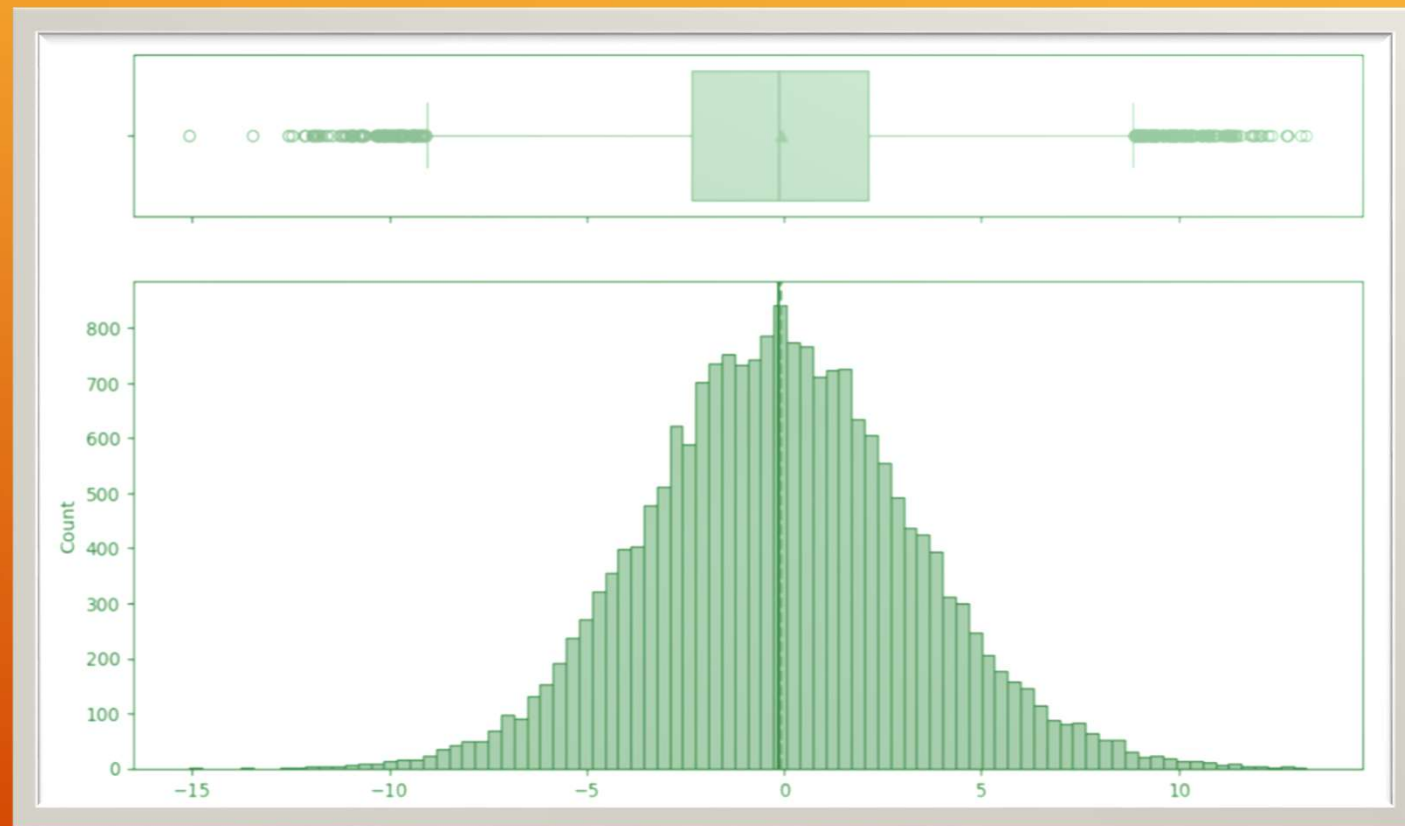
- Statistical Summary of the Dataset showed a wide range between negative and positive between Mean and Max.
- V3 & V35 had the highest positive values, which says we have Outliers in the data

	count	mean	std	min	25%	50%	75%	max
V1	19982.000	-0.272	3.442	-11.876	-2.737	-0.748	1.840	15.493
V2	19982.000	0.440	3.151	-12.320	-1.641	0.472	2.544	13.089
V3	20000.000	2.485	3.389	-10.708	0.207	2.256	4.566	17.091
V4	20000.000	-0.083	3.432	-15.082	-2.348	-0.135	2.131	13.236
V5	20000.000	-0.054	2.105	-8.603	-1.536	-0.102	1.340	8.134
V6	20000.000	-0.995	2.041	-10.227	-2.347	-1.001	0.380	6.976
V7	20000.000	-0.879	1.762	-7.950	-2.031	-0.917	0.224	8.006
V8	20000.000	-0.548	3.296	-15.658	-2.643	-0.389	1.723	11.679
V9	20000.000	-0.017	2.161	-8.596	-1.495	-0.068	1.409	8.138
V10	20000.000	-0.013	2.193	-9.854	-1.411	0.101	1.477	8.108
V11	20000.000	-1.895	3.124	-14.832	-3.922	-1.921	0.119	11.826
V12	20000.000	1.605	2.930	-12.948	-0.397	1.508	3.571	15.081
V13	20000.000	1.580	2.875	-13.228	-0.224	1.637	3.460	15.420
V14	20000.000	-0.951	1.790	-7.739	-2.171	-0.957	0.271	5.671
V15	20000.000	-2.415	3.355	-16.417	-4.415	-2.383	-0.359	12.246

Exploratory Data Analysis - Univariate

Plotting the Data on Histograms and Boxplots for all the Variables showed there were many Outliers. Below is an example of one of the Diagrams.

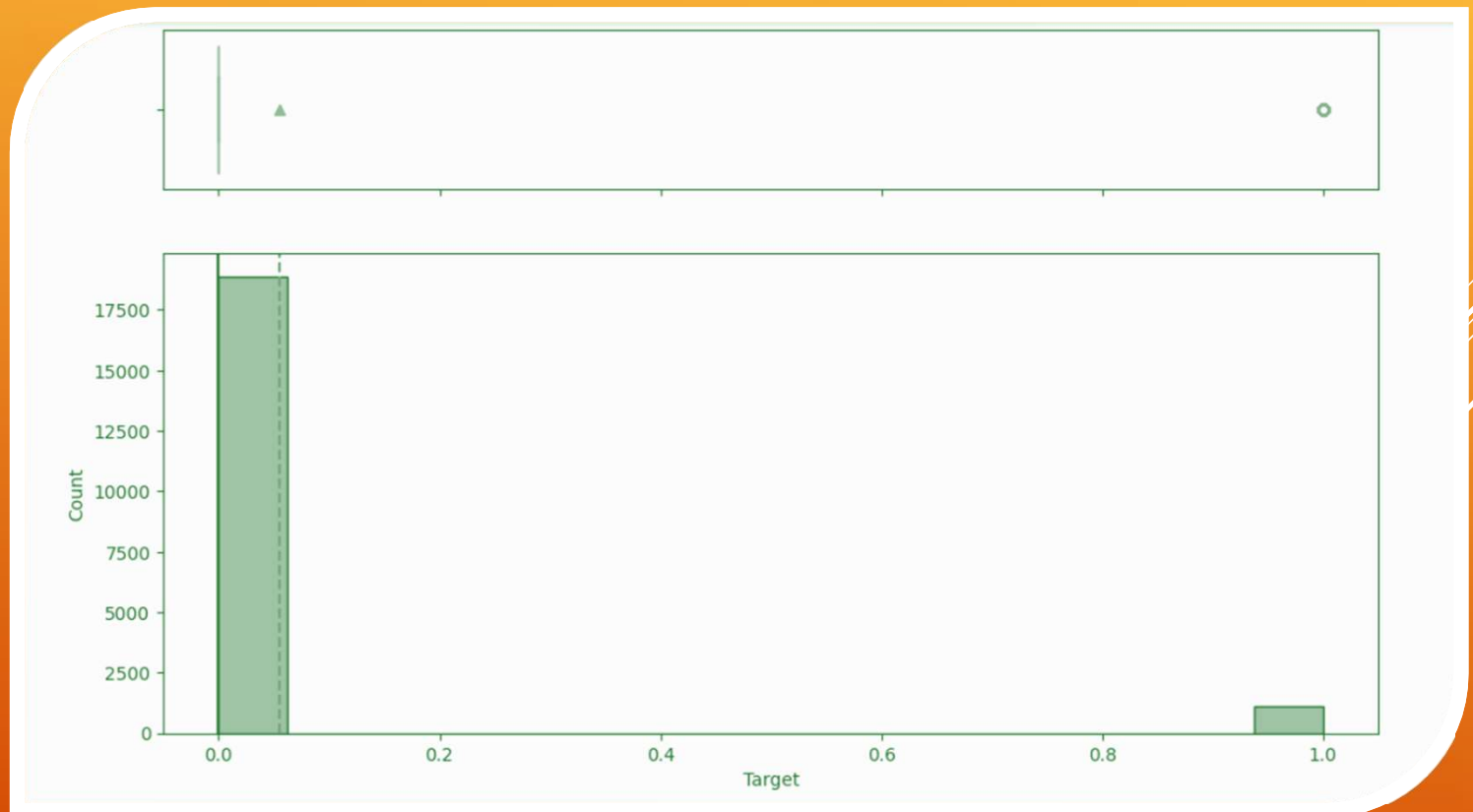
- V4 output Plots - Mean was very much inline with the actual value.
- This output shows many Outliers and the plots for the remaining 39 Variables showed this as well.



Exploratory Data Analysis - Univariate

Target Variable Analysis = Imbalanced Data

- Train Data
 - 94% No Failures
 - -1% Failures
- Test Data
 - Matching Analysis



Data Pre-Processing

1.

- Train & Test already separated
- Validation Set split from Train Data

2.

- Created matching amounts for all three sets
- 5,000 Rows & 40Variables

3.

- Missing Values were Imputed as Median Value
- No Data Leakage

Model Criterion & Build Explanation

❑ Classification Model

- True Positives (TP) - Correct Model Predictions {Repair Costs}
- False Negatives(FN) - Actual Failures with no model Prediction {Replacement Costs}
- False Positives(FP) - Detected Failure - but did not fail {Inspection Costs}

❑ Optimizing the “Recall” Metric

- The model correctly predicts the maximum number generator failures
- Greater the Recall amount, the greater the chances of minimizing False Negatives

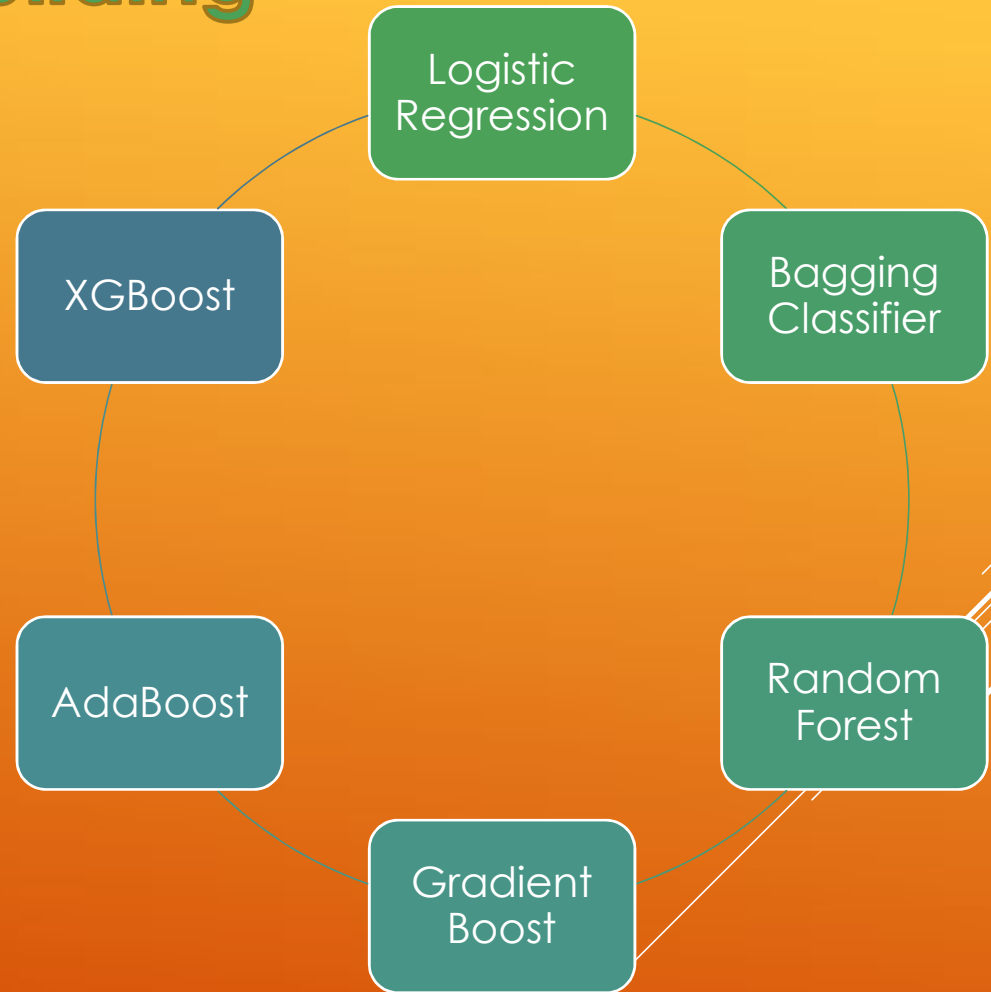
❑ False Negatives need to be at a minimum, or there will be increased maintenance costs

❑ Accuracy - Precision - F1 Score was also computed

❑ Scorer was used in Cross-Validation and Hyper Parameter Tuning of all models

Model Building

- Train and Validation Analysis Only
- Built on Original Data
- Cross-Validation - Stratify - Kfold
- Boxplots for CV Scores & Algorithm Comparison
- Oversampling & Undersampling Data
- Grid Search & Random Search
- Hyper Parameter Tuning
- Performance Comparison, Final Model & Test Data Performance
- Feature Importance's
- Pipeline for Final Model



Model Build - Original Data

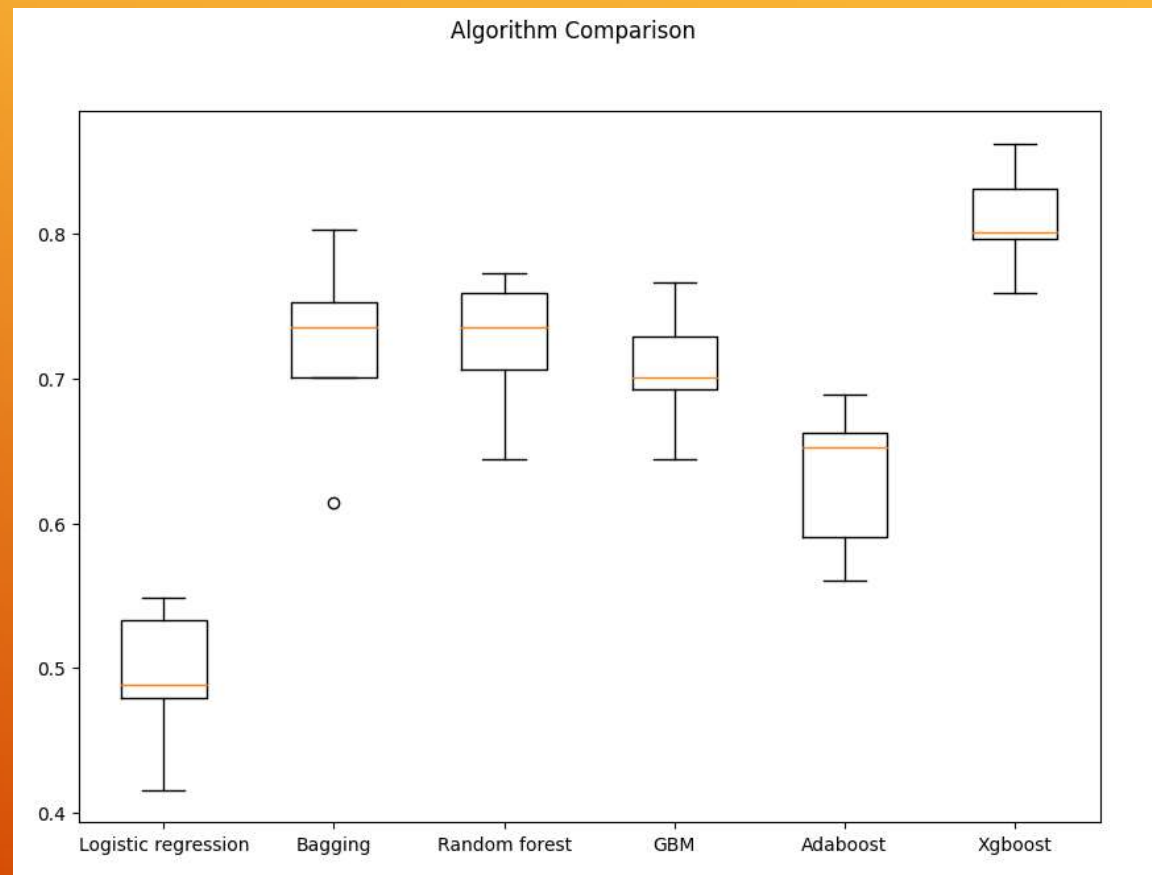
- XGBoost has given the best Cross-Validation Recall
Train: 81 Validation: 83
- Bagging offered the next highest Recall Score
Train: 72 Validation: 73

Cross-Validation performance on training dataset:

Logistic regression: 0.4927566553639709
Bagging: 0.7210807301060529
Random forest: 0.7235192266070268
GBM: 0.7066661857008874
Adaboost: 0.6309140754635308
Xgboost: 0.8100497799581561

Validation Performance:

Logistic regression: 0.48201438848920863
Bagging: 0.7302158273381295
Random forest: 0.7266187050359713
GBM: 0.7230215827338129
Adaboost: 0.6762589928057554
Xgboost: 0.8309352517985612



Oversampled Data

Before OverSampling, counts of label '1': 832
Before OverSampling, counts of label '0': 14168

After OverSampling, counts of label '1': 14168
After OverSampling, counts of label '0': 14168

After OverSampling, the shape of train_X: (28336, 40)
After OverSampling, the shape of train_y: (28336,)

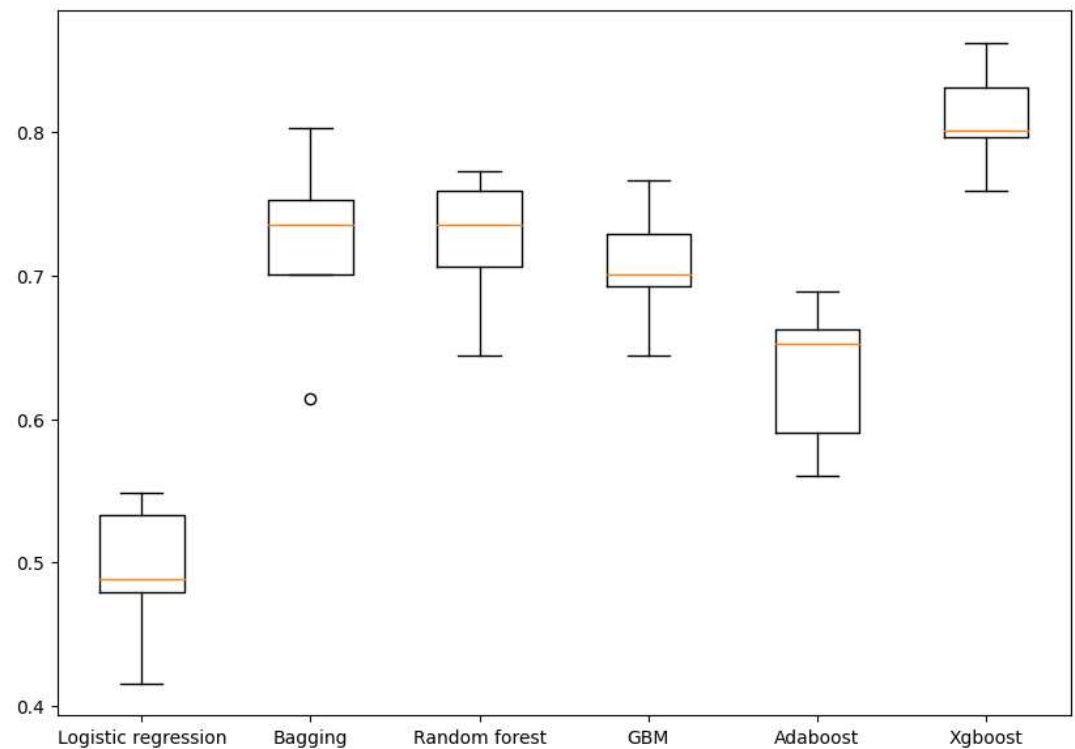
Cross-Validation performance on training dataset:

Logistic regression: 0.4927566553639709
Bagging: 0.7210807301060529
Random forest: 0.7235192266070268
GBM: 0.7066661857008874
Adaboost: 0.6309140754635308
Xgboost: 0.8100497799581561

Validation Performance:

Logistic regression: 0.48201438848920863
Bagging: 0.7302158273381295
Random forest: 0.7266187050359713
GBM: 0.7230215827338129
Adaboost: 0.6762589928057554
Xgboost: 0.8309352517985612

Algorithm Comparison



Undersampled Data

Before UnderSampling, counts of label '1': 832
Before UnderSampling, counts of label '0': 14168

After UnderSampling, counts of label '1': 832
After UnderSampling, counts of label '0': 832

After UnderSampling, the shape of train_X: (1664, 40)
After UnderSampling, the shape of train_y: (1664,)

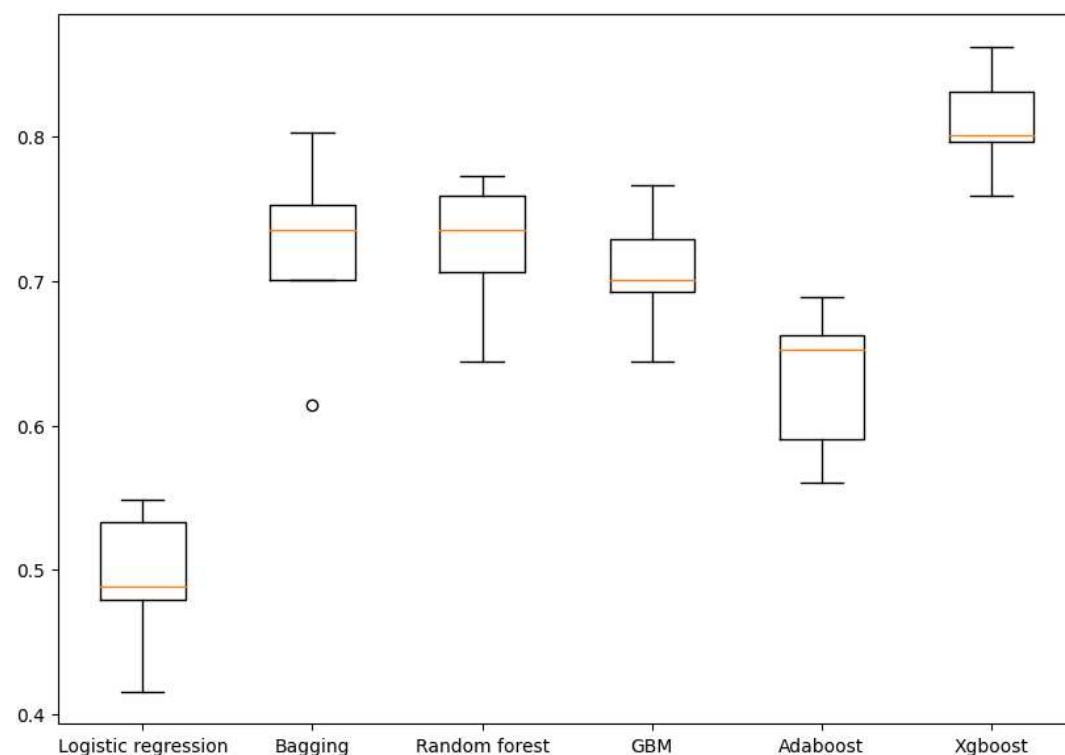
Cross-Validation performance on training dataset:

Logistic regression: 0.4927566553639709
Bagging: 0.7210807301060529
Random forest: 0.7235192266070268
GBM: 0.7066661857008874
Adaboost: 0.6309140754635308
Xgboost: 0.8100497799581561

Validation Performance:

Logistic regression: 0.48201438848920863
Bagging: 0.7302158273381295
Random forest: 0.7266187050359713
GBM: 0.7230215827338129
Adaboost: 0.6762589928057554
Xgboost: 0.8309352517985612

Algorithm Comparison



Hyper Parameter Tuned Models - Recall

PERFORMANCE - TRAIN vs. VALIDATION

Random Undersampled Data
Train: 93 Validation: 88

XGBoost Oversampled Data
Train: 1.00 Validation: 89

Gradient Boost Oversampled Data
Train: 99 Validation: 85

Best Models

Random
Undersampled

XGBoost
Oversampled

Gradient Boost
Oversampled

Comparison - Model Performance

Gradient Boost:
Random
Search

AdaBoost:
Random
Search

Random Forest:
Random
Search

XGBoost:
Random
Search

Comparison - Model Performance

Training performance comparison:

	Gradient Boosting tuned with random search	AdaBoost classifier tuned with random search	Random forest tuned with random search	XGBoost tuned with random search
Accuracy	0.993	0.958	0.961	0.994
Recall	0.992	0.916	0.933	1.000
Precision	0.994	1.000	0.989	0.988
F1	0.993	0.956	0.960	0.994

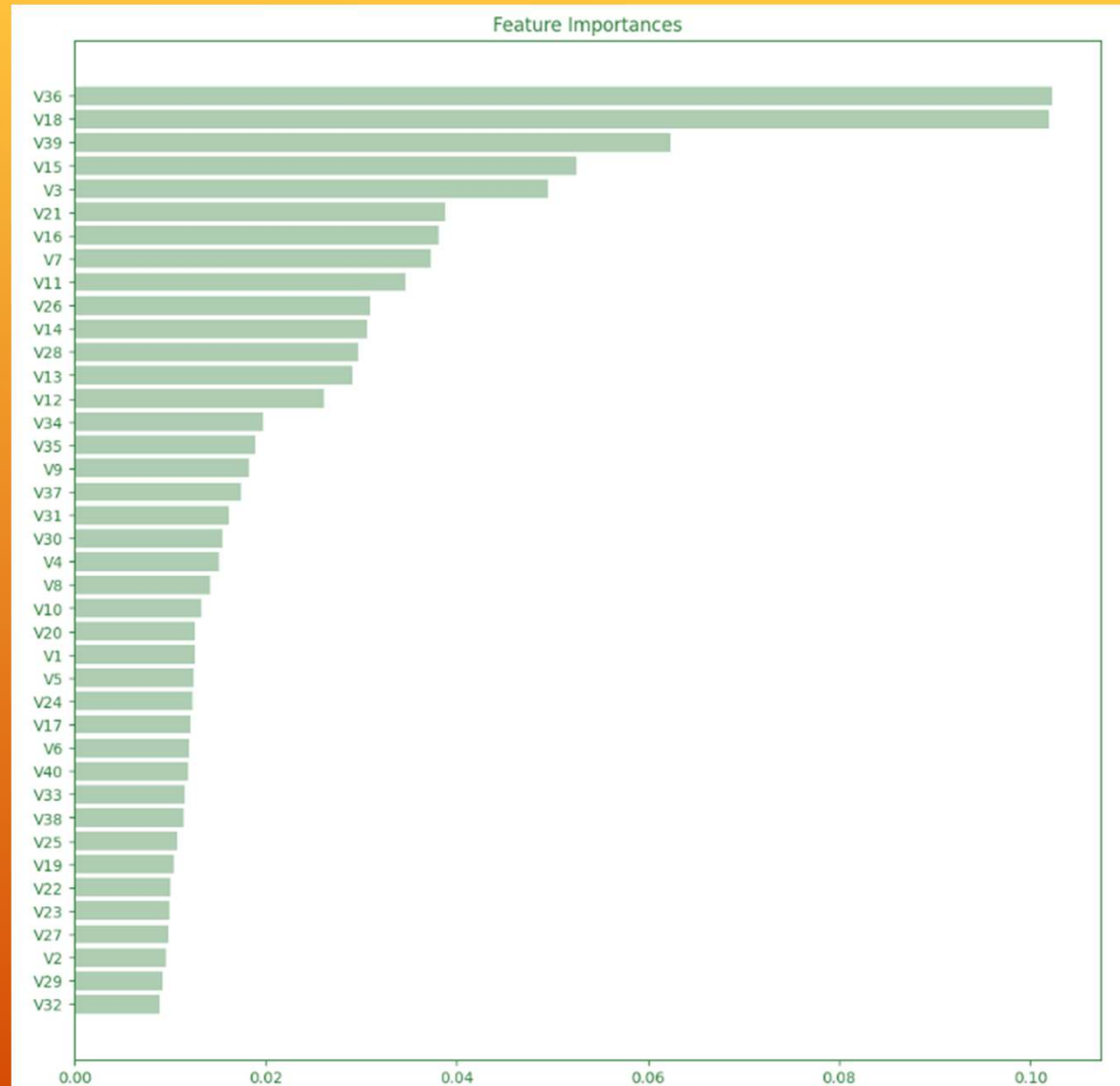
- Random Forest was chosen as the Final Model because of the comparison of Train vs. Validation set Performance
- XGBoost Train was too good to be true with a Recall of 1.00 and Validation going down to 89

Validation performance comparison:

	Gradient Boosting tuned with random search	AdaBoost classifier tuned with random search	Random forest tuned with random search	XGBoost tuned with random search
Accuracy	0.969	0.985	0.938	0.967
Recall	0.856	0.784	0.885	0.892
Precision	0.678	0.928	0.468	0.648
F1	0.757	0.850	0.612	0.750

Random Forest Tuned with Random Search

- The features are sensor data based on the Final Model
- The characteristics of sensors V36 & V18 show that their performance should be replicated for all other sensors
- Random Search CV uses a random grid search technique to compute hyper parameter values



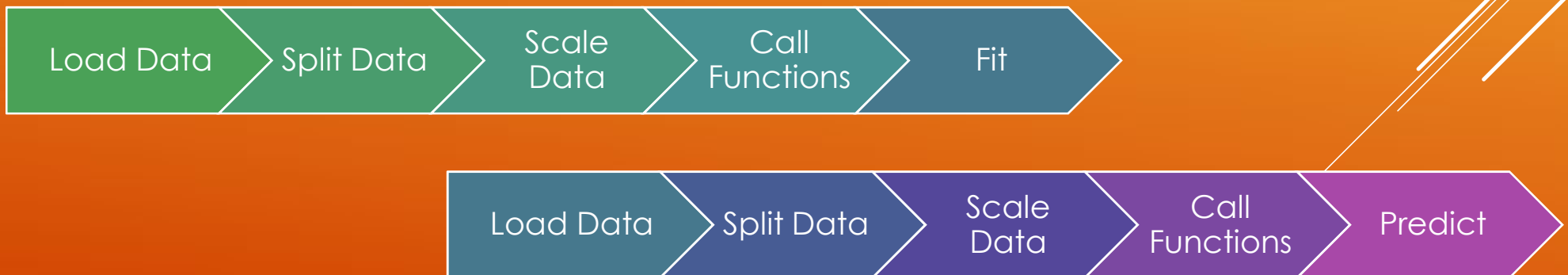
Tuned Random Forest - Undersampled Data

- This model had the best result after running on Test Data
- Oversampled Data was less than 30%, and Undersampled Data was 45%
- This still isn't a great result, but more information is required over time to produce better results with more predictors
- With this information we have we will build a Pipeline in order to Productionize this model to run on future Test Data

	Accuracy	Recall	Precision	F1
0	0.726	0.457	0.989	0.625

Pipeline for Production Model

- Random Forest - Random Grid Search - Undersampled Data was the Best Model
- Creation of a Pipeline ensures that sequentially applying steps that use Transform, Fit, Estimator functions helps to ensure a standardized model that can be used repeatedly by different users on Test Data
- Utilizing these steps creates the pipeline. Specifying names is the most crucial part of what makes the pipeline obsolete and this consistency in naming reduces errors by other users when implementing a Pipeline Model



Insights:

- The Key Features of V36 & V18 are the highest and those specifically should be evaluated more closely to see what makes them stand out amongst all of the other features
- Univariate Analysis showed similar results in our data, and all of our Histograms (40) showed similar Bell Curves with Median values very close to the actual value. There were many Outliers on both ends of the spectrum by some skewing on either end and also tails toward positive and negative
- In this Data Set were given Cyphered values from sensors fitted across different machinery within the process of creating Wind Energy, and we should keep this in mind for future analysis

Conclusions:

- Our final model did not give overwhelmingly great results, but it offered the best processing of the data and will get better with the more data that is run through it
- We need more predictor variables that are reflect sensors on either machinery or environmental factors. The data in this analysis was mixed and offered a wide spectrum of positive and negative values leading to numerous outliers