ReneWind - Problem Statement

PGP - Data Science & Business Analytics
January 10, 2024

Leslieane Beltran

Contents/Agenda

- Executive Summary
- Business Overview/Solution Approach
- EDA Results
- Data Preprocessing
- Model Performance Summary for hyperparameter tuning
- Model building with Pipeline

Executive Summary

- Wind energy is key to cutting environmental impact, and predictive maintenance helps keep turbines running smoothly.
- ReneWind is using machine learning to predict generator failures and lower maintenance costs.
- The data includes 40 features and 25,000 records (20,000 for training, 5,000 for testing), all anonymized for privacy.
- The focus is on reducing expensive replacements by catching issues early with repairs or inspections.
- Models will classify outcomes as failures or no failures, helping prioritize maintenance efforts.
- The goal is to boost efficiency and reliability in wind power operations.

Business Problem Overview

- Wind turbines are critical to renewable energy, but generator failures increase maintenance costs and downtime.
- Predictive maintenance can help prevent failures by using sensor data to identify issues before they occur.
- ReneWind collects data from turbine sensors, including environmental factors and machine performance metrics.
- Failures lead to the need for replacements, repairs and inspections which are costly.
- ReneWind needs an accurate machine learning model to predict failures, which will create proactive maintenance strategies.
- Lowering failures and optimizing maintenance will lower costs and improve turbine efficiency and reliability.

Solution Approach

The following describe the solution approach:

- Collect and preprocess data, ensuring quality and readiness for model training.
- Analyze the data to identify patterns and trends when it comes to turbine failure.
- Develop different models.
- Fine-tune the model settings to get the best results while keeping accuracy and costs in mind.
- Check how the models perform, making sure to catch as many real failures as possible to avoid expensive replacements.
- Use the best model to predict failures in real time and help plan maintenance ahead of time.

EDA Results - Univariate Analysis

Key Results

- Most variables have a fairly normal, balanced distribution, though some lean slightly to the positive side.
- There are outliers in all of them, especially on the higher end, which could point to rare or unusual events that might help predict failures.
- The data looks stable overall, with the mean and median lining up nicely for most features.
- While most values cluster around the center, those outliers and tails could hold valuable clues for improving predictions and identifying potential issues early.

Values in Target Variable

Train Data

Target	count
0	18890
1	1110

dtype: int64

Test Data

Target	count
0	4718
1	282

dtype: int64

Data Preprocessing

Splitting data into training and validation set:

```
(15000, 40) (5000, 40) (5000, 40)
```

Data was split into a training set with 15,000 samples, a validation set with 5,000 samples, and a separate test set with 5,000 samples

- Zero duplicate values
- Missing values were treated to ensure no gaps remained in the training, validation, or test sets, which is crucial for machine learning models to perform accurately.

Hyperparameter Tuning

AdaBoost (oversampled Data)

Model does great on the oversampled training data with high accuracy, recall, and precision, but not good on validation set, with precision and F1 score dropping a lot, which suggests it might be overfitting and not generalizing well.

	Accuracy	Recall	Precision	F1
0	0.936	0.913	0.957	0.934

	Accuracy	Recall	Precision	F1
0	0.947	0.871	0.513	0.645

Random Forest (Undersampled Data)

The model does great on the undersampled training data with near-perfect scores. Performance drops on the validation set, especially in recall and F1 score, which suggests it might be overfitting and not generalizing well to new data.

	Accuracy	Recall	Precision	F1
0	0.999	1.000	0.999	0.999

	Accuracy	Recall	Precision	F1
0	0.988	0.842	0.944	0.890

Gradient Boosting (Oversampled Data)

The model does really well on the oversampled training data, but its performance takes a hit on the validation set, especially in precision and overall balance, which shows it might be overfitting and struggling to handle new data

	Accuracy	Recall	Precision	F1
0	0.967	0.947	0.987	0.966

	Accuracy	Recall	Precision	F1
0	0.975	0.881	0.723	0.794

Model Performance Summary

```
Training performance comparison:
```

	Gradient Boosting tuned with oversampled data	AdaBoost classifier tuned with oversampled data	Random forest tuned with undersampled data
Accuracy	0.967	0.936	0.999
Recall	0.947	0.913	1.000
Precision	0.987	0.957	0.999
F1	0.966	0.934	0.999

The Random Forest model is the best model according to the table above. It ensures ReneWind catches all failures (avoiding costly replacements) while keeping unnecessary inspections low, making it perfect for their mission to lower costs and boost turbine efficiency.

Model Building with Pipeline

Steps

- Set up a pipeline with the tuned Random Forest model to handle everything in one go.
- Filled in missing values using a SimpleImputer with the median to keep things clean.
- Balanced the data with Random Under Sampling (RUS) so the model could focus on both failures and non-failures.
- Trained the pipeline on the data and tested it to see how well it performs.

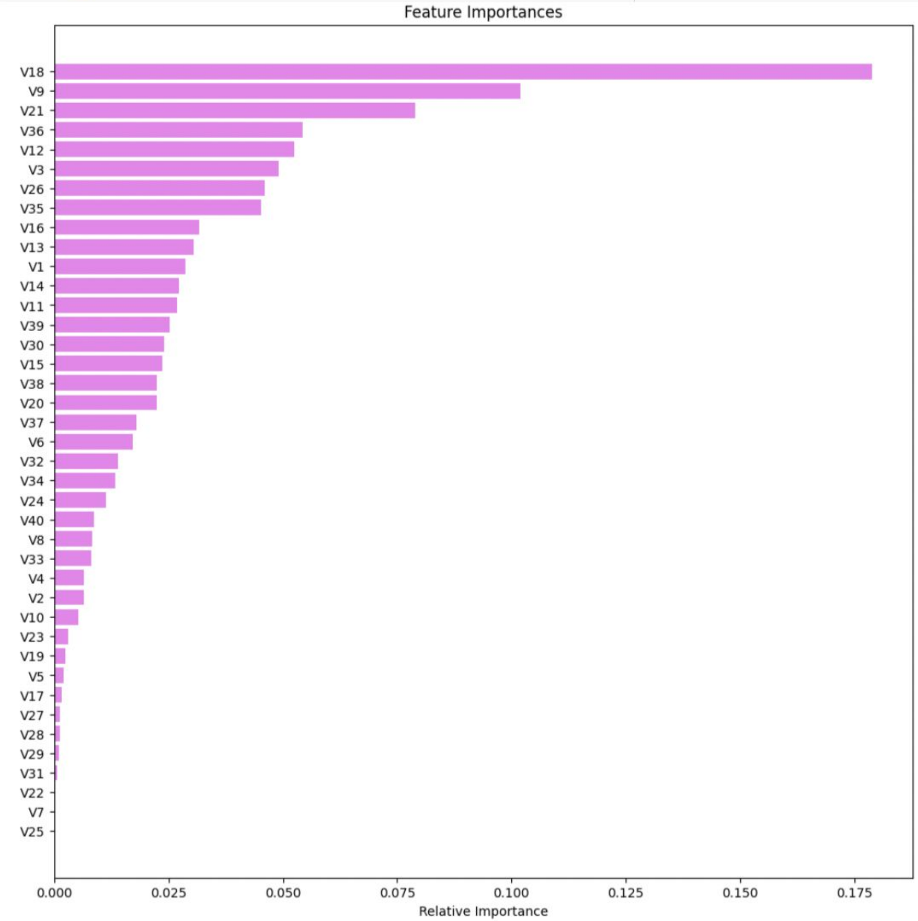
Summary of Model

	Accuracy	Recall	Precision	F1
0	0.982	0.688	0.985	0.810

The model does a great job overall with 98.2% accuracy and solid precision (0.985), but it could be better at catching all failures since the recall is a bit low (0.688).

Important Factors

- **V18, V9, and V21** are the top features
- These features probably capture important signals or patterns related to turbine performance.



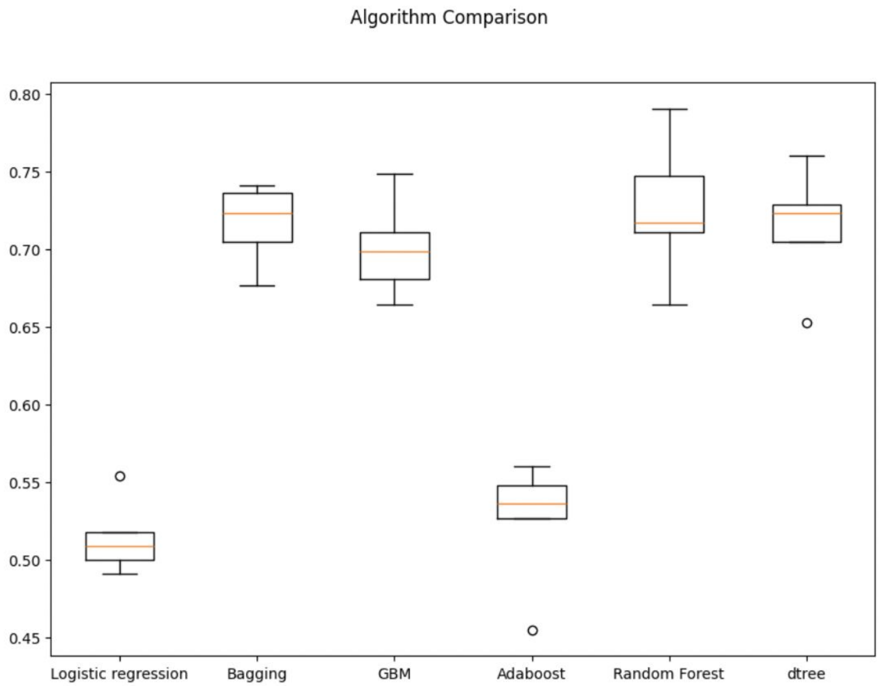
Insights & Conclusions

Insights & Conclusions

- A machine learning model was created to help cut down the maintenance costs for wind energy machinery and processes.
- Between the Gradient Boosting (with oversampled data), AdaBoost (with oversampled data), and Random Forest (with undersampled data) models, the Random Forest model was picked because it had the best accuracy, precision, recall, and F1 score.
- A pipeline was also built to make it easy to use the final model in production.
- This model is expected to do the best job of predicting failures and non-failures, helping prioritize what needs to be fixed. By doing so, it can reduce breakdowns, streamline maintenance, lower costs, and improve turbine performance.
- The most important factors for predicting failures were identified as "V18," "V9," and "V21." Knowing this can help focus on collecting more detailed sensor data to make the model even better and save more on maintenance costs.

Appendix

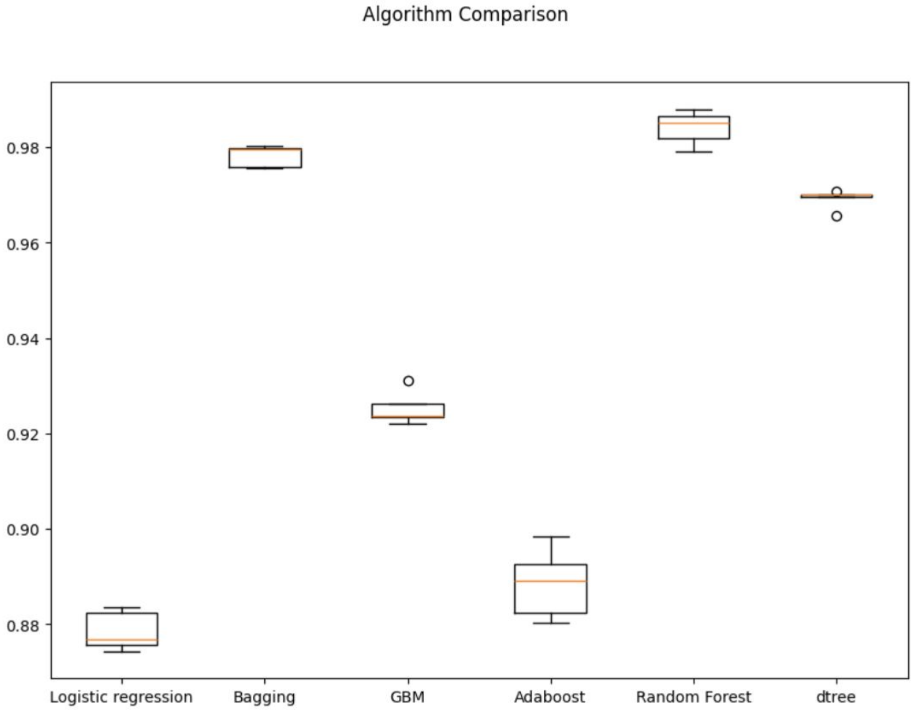
Model Performance Summary - Original Data




```
Cross-Validation Cost:

Logistic regression: 0.5144578313253012
Bagging: 0.7163696702979583
GBM: 0.7007070196955487
Adaboost: 0.525322848279345
Random Forest: 0.7259577231080009
dtree: 0.7139600317437415

Validation Performance:

Logistic regression: 0.420863309352518
Bagging: 0.6870503597122302
GBM: 0.6654676258992805
Adaboost: 0.46402877697841727
Random Forest: 0.6906474820143885
dtree: 0.697841726618705
```

Random Forest is the best all-around model—strong, consistent, and reliable. Bagging or GBM could also work well if you're looking for slightly different trade-offs, but Logistic Regression and Adaboost don’t seem like great fits.

Model Performance Summary - Oversampled Data

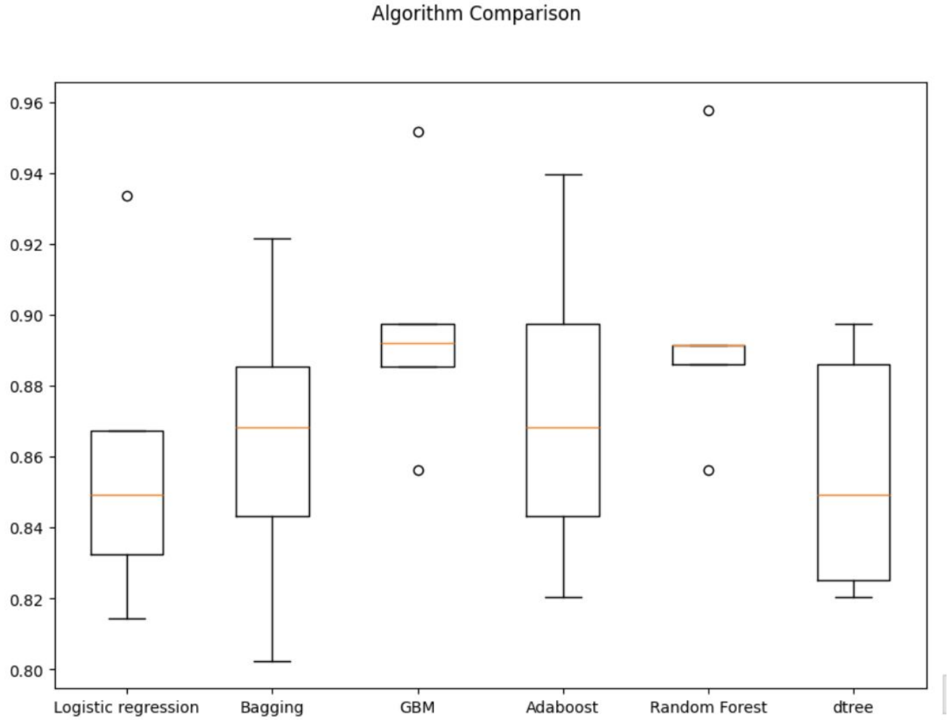



```
Cross-Validation Cost:

Logistic regression: 0.8785997074005053
Bagging: 0.97826082407636
GBM: 0.9253246282534132
Adaboost: 0.8886220995072442
Random Forest: 0.9841897377938856
dtree: 0.9692265593453107

Validation Performance:

Logistic regression: 0.8381294964028777
Bagging: 0.802158273381295
Adaboost: 0.8237410071942446
Random Forest: 0.8201438848920863
dtree: 0.7769784172661871
```

Adaboost looks like the best option. Balances good validation performance and efficiency. Random Forest and Bagging are solid too, but their higher costs make Adaboost stand out for oversampled data.

Model Performance Summary - Undersampled Data

```
Cross-Validation performance on training dataset:

Logistic regression: 0.8594617992929804
Bagging: 0.8642522184546569
GBM: 0.8966885506096242
Adaboost: 0.8738691292114567
Random Forest: 0.8966957650963133
dtree: 0.8557751965947624

Validation Performance:

Logistic regression: 0.8453237410071942
Bagging: 0.8633093525179856
GBM: 0.8812949640287769
Adaboost: 0.8489208633093526
Random Forest: 0.8812949640287769
dtree: 0.8273381294964028
```



GBM and Random Forest are the best choices, both show strong and consistent performance. Bagging and Adaboost are good alternatives but slightly less reliable. Logistic Regression and Decision Tree don't perform as well with undersampled data.