Renewind

Model Tuning and PGP-DSBA

10/08/2023

Contents / Agenda

- Executive Summary
- Business Problem Overview and Solution Approach
- EDA Results
- Data Preprocessing
- Model performance summary for hyperparameter tuning.
- Model building with pipeline

Executive Summary

- Insights & Recommendations:-

- A machine learning model has been built to minimize the total maintenance cost of machinery/processes used for wind energy production
- The final tuned model (XGBoost) was chosen after building ~7 different machine learning algorithms & and further optimizing for target class imbalance (having few "failures" and many "no failures" in the dataset) as well as finetuning the algorithm performance (hyperparameter and cross-validation techniques)
- A pipeline was additionally built to produce the final chosen model
- The main attributes of importance for predicting failures vs. no failures were found to be "V18", "V39", "V26", "V3" & and "V10" in order of decreasing importance. This added knowledge can be used to refine the process of collecting more frequent sensor information to be used in improving the machine learning model to further decrease maintenance costs.
- Features V36, 16, and 18 have the most impact on determining whether a windmill will fail or not
- The company should focus on improving those features so they can reduce the number of failures and the amount of money spent on repairs and replacements.

Business Problem Overview and Solution Approach

- ReneWind” is a company working on improving the machinery/processes involved in the production of wind energy using machine learning and has collected data on generator failure of wind turbines using sensors.
- The objective is to build various classification models, tune them, and find the best one that will help identify failures so that the generators can be repaired before failing/breaking to reduce the overall maintenance cost.
- The company has provided a data description which includes the CSV files of train and test data.

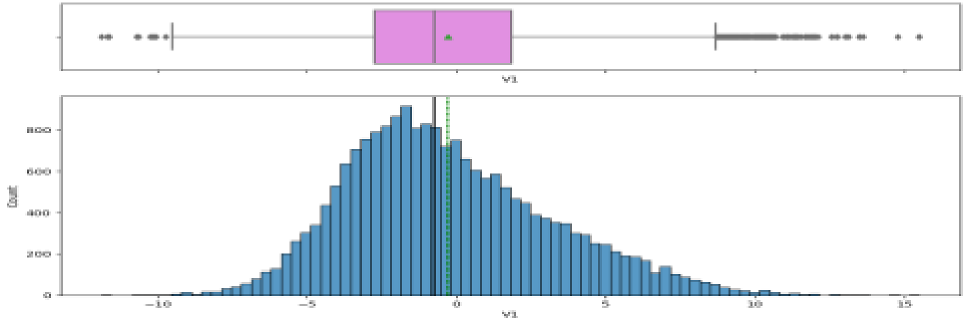
EDA Results

- There are 2 kinds of data that are going to see the data set overview which includes train data as well as test data.
- There are approx. 5000 rows and 41 attributes in the dataset of training data.
- All variables in the training set are numerical and we can observe that there are some missing values in V1 and v2.
- Almost there are the float-type numbers. There are approx. 46 missing values for v1 and 39 missing values for v2.
- There are no duplicate values in the dataset.
- If we check about the statistical summary of data which shows there is the spread of attributes which can be solved in the univariate analysis.

Link to Appendix slide on data background check

EDA Results

Univariate analysis:-



Observations:-

- The above graph is an example of how I have made the boxplot and histogram for each attribute from v1 to v40 and I have observed that all independent values are outliners, and the target variable is suffering from an imbalanced distribution where 0 is 94% and 1 is 5%.
- It also includes the positive and negative outliers .

Data Pre-Processing

- Firstly, to avoid any leakages we will first split the dataset in the training file to train and validate.
- After splitting the data, we have 16000 rows with 40 validation in train data a 4000 rows with 40 of validation in test data.
- We have the split of 75:25 split between the train and validation data.
- If we divide the test data, then it shows that there is the same number of rows in the test data as there is in the validation data.

Missing value imputation:-

- We will use the median to impute missing values in V1 and V2 columns.
- From the codes and everything we get to know that we do not have any missing values in the training, value, or test sets.

Model Building

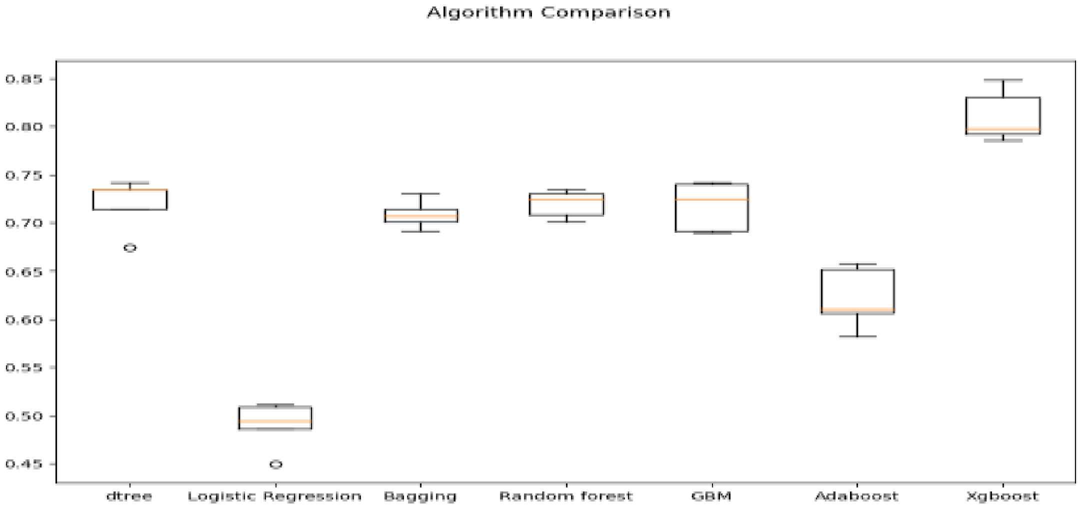
- Here shows the validation performance of all models:-

```
Cross-Validation performance on training dataset:

dtree: 0.7196280873636767
Logistic Regression: 0.48086012924522133
Bagging: 0.7883222241382213
Random forest: 0.7195099193884254
GBM: 0.7173538805719928
Adaboost: 0.6215641465117756
Xgboost: 0.8108842912488112

Validation Performance:

dtree: 0.7287287287287287
Logistic Regression: 0.49090909090909097
Bagging: 0.7387387387387387
Random forest: 0.7432432432432433
GBM: 0.7432432432432432
Adaboost: 0.6576576576576577
Xgboost: 0.8153153153153153
```

Observation:-

- The cross-validation training performance scores (customized metric) are like the validation performance score. This indicates that the default algorithms on the original dataset can generalize well.
- There is a tendency for some models (decision tree, random forest, bagging, and XGBoost) to overfit the training set; as the training performance score (customized metric) approaches 1

Model Building



•XGBoost and Random Forest (~ 0.71) have the best average (& and median) training cross-validation scores (on the customized metric). This is closely followed by the Bagging Classifier (~ 0.68)
•XGBoost and AdaBoost each have one outlier as can be observed from the boxplot
•The boxplot widths (spread of CV scores) are small for XGBoost, Random Forest, and Bagging Classifier as well, indicating these are reliable models to choose for further optimization

Model Building and Oversampled and Undersampled Training Data

Before Oversampling, counts of label '1': 888
Before Oversampling, counts of label '0': 15112
After Oversampling, counts of label '1': 15112
After Oversampling, counts of label '0': 15112
After Oversampling, the shape of train_X: (30224, 40)
After Oversampling, the shape of train_y: (30224,)

Observation:-
Xgboost is the best performer, followed by GBM and random forest, and dtree.

Model Building and Oversampled and Undersampled Training Data

Before Under sampling, counts of label '1': 888
Before Under sampling, counts of label '0': 15112
After Under sampling, counts of label '1': 15112
After Under sampling, counts of label '0': 15112
After Under sampling, the shape of train_X: (1776,40)
After Under sampling, the shape of train_y: (1776)

Observation:-
Xgboost is the best performer, followed by GBM random forest, and dtree.

Hyperparameter Tuning

- The best hyperparameters using RandomizedSearch CV for XGBoost model were found.
- The average cross-validation training performance score (customized metric) using the best parameter XGBoost model is 0.80. This is similar to the performance score (customized metric) on the validation set i.e., 0.82. This indicates the model may generalize with a performance score of ~0.80-0.82
- The model does however tend to overfit the training set as can be observed from training performance (customized metric score of 0.998)
- With the Gradient Boost Model, the recall on the training set is .985 and drops to .842 on the validation set meaning it's able to identify 84.2% of the failures in the validation set as failures.

Link to Appendix slide on model assumptions

Final Model.

Model Performance comparison and choosing the final model:

```
Training performance comparison:
```

	xgb_oversampled_train	gb_oversampled_train	ada_oversampled_train	rf_oversampled_train	dtree_oversampled_train
Accuracy	0.966616	0.821665	0.941867	0.903886	0.722571
Recall	0.934026	0.658020	0.883735	0.807372	0.449444
Precision	0.999151	0.978163	1.000000	1.000000	0.990521
F1	0.965491	0.786771	0.938279	0.893421	0.618326

```
Validation performance comparison:
```

	xgb_oversampled_val	gb_oversampled_val	ada_oversampled_val	rf_oversampled_val	dtree_oversampled_val	lrg_overs
Accuracy	0.988500	0.959250	0.985000	0.983750	0.965000	0.968750
Recall	0.855856	0.644144	0.770270	0.720721	0.463964	0.490991
Precision	0.931373	0.629956	0.950000	0.981595	0.830645	0.844861
F1	0.892019	0.636971	0.850746	0.831169	0.595378	0.621083

- It seems that XGB on oversampling is the best performance with a recall of 0.85¶

Pipelines to build the final model:-

- The pipeline performance is as expected indicating it was built accurately to replicate the final chosen model after necessary pre-processing

Happy Learning !