ReneWind Renewable Energy

Model Tuning

Mona Desai
Date:02/05/2023

Contents / Agenda

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

Executive Summary

- **Conclusion, Actionable insights**
  - On the given data, all 4 (tuned) diff models performed well almost equally - high value of Recall, Accuracy and Precision values
  - ADA Booster performed more consistently across all data sets and all scores are comparable between Train, Validation and Test set
  - Our Model had a very high Accuracy (), Recall (), Precision () leading ultimately to reduced Repair, Replacement & Inspection costs.
  - Among all the features, Variables V30, V9, V18 and V12 impacts the most on the prediction of the Failure.
  - Both given Train and Test data, the number of Failure is ~5.5 %
  - The Validation Recall for Test set after tuning and finalizing the Adaboost model is **0.84** with Accuracy **0.97** and Precision **0.78** which is **very comparable** with with the results on test sets build using Pipeline **0.85, 0.97, 0.77**
- **Recommendations**
  - Identifying combination of variable values leading to highest failures
  - Speed up model run performance. Currently, models run slow.
  - Identify and Eliminate with much less impact on 'Target' - This simplifies the model and run times
  - Make sure the Test Data should only be used to check the final tuned model to avoid the Data Leakage.

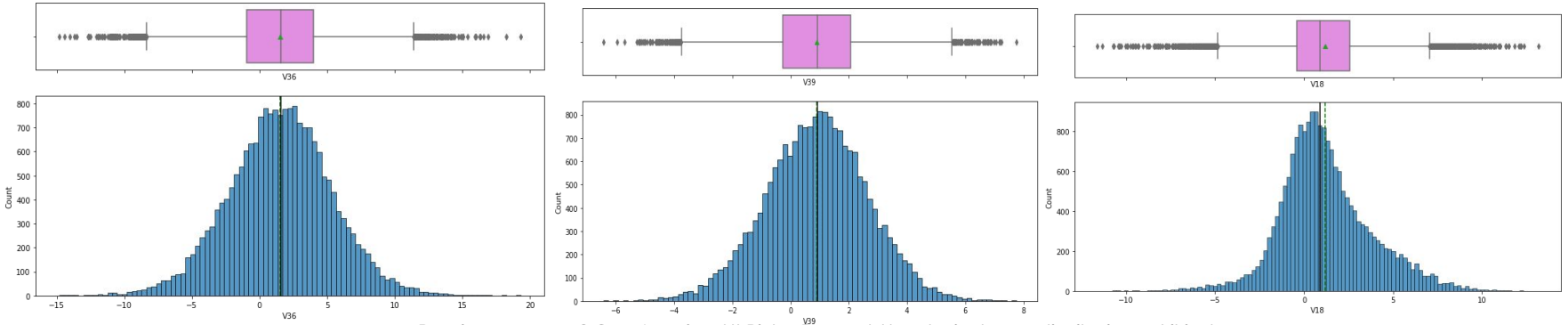
Business Problem Overview and Solution Approach

- **Problem**
  - Finding the best fit and tuned model to predict the failures so the generators can be repaired before breaking
  - We need to choose the metric which will ensure that the maximum number of generator failures are predicted correctly by the model.
  - We want Recall to be maximized as greater the Recall, the higher the chances of minimizing false negatives.
  - We want to minimize false negatives because if a model predicts that a machine will have no failure when there will be a failure, it will increase the maintenance cost.

- **Solution approach / methodology**
  - Exploratory Data Analysis
    - Overview of dataset, Univariate analysis
    - Split Train set into Train and Validation sets
  - Model Building
    - K-Fold and Cross validation score for recall on Original, Oversampled and Undersampled data to find the best models for the hyper parameter tuning
  - Hyperparameter Tuning for the chosen Models AdaBoost,Random Forest, Gradient Boosting and XGBoost
  - Created Pipeline using the best parameters
  - Model performance comparison of the Tuned Models comparing Recall scores, Accuracy, Precision and F1 scores overall.
  - Use Pipelines to build the final model
  - Missing Value Treatment

EDA Results

- **Key results from EDA**
  - The data contains 40 different variables (sensors) predicting failure of the wind turbines.
  - 2 given data sets named, “Train” and “Test”
  - Train set with 20000 rows and 41 columns
  - Test set with 5000 rows and 41 columns
  - All fields are numeric fields
  - Statistical summary suggests, values in all columns contains a wide range of -ve in ~15’s to +ve ~15’s
  - Both Train and Test data has ~5.5% “Target” variable 1, which predicts - Machine Failure
  - From the “Boxplot” and “Histogram”, all 40 variables are almost “Normally” distributed



Data Preprocessing

- **Duplicate value check and Missing value treatment**
  - No duplicate values however column V1 and V2 variables have some missing values in both train and test sets.
  - Imputed missing values with “Median” in all Train, Validation and Test sets after splitting the datasets to avoid variations and Data leakage.
- **Outlier check (treatment if needed)**
  - All three numerical columns has outliers, however we will not treat them as they are proper values
- **Feature engineering and Data preparation for modeling**
  - **“**Target” is the dependent variable with “1 - Failure” and “0- No Failure”
  - We are given Train and Test sets readily before hand
  - We drop “Target” variable from both Train and Test sets separately
  - We further spit Train set in to Train and Validation sets into 75:25 part
  - Build functions to calculate different metrics and confusion matrix to use the same code for each model
  - To maximize Recall, we can use Recall as a scorer in cross-validation and hyperparameter tuning.

Model Performance Summary

- **Summary of performance metrics for training and validation data for tuned models**
  - After looking at performance of all the models, chose AdaBoost using Oversampled,Random Forest using Undersampled,Gradient Boosting using Oversampled and XGBoost using Oversampled data models for the further improve with hyperparameter tuning.
  - random_state, n_estimators, learning_rate, base_estimators, max_featres are more often used among many hypertuning parameters used to tuned the different models
  - Performance of all tuned models on Train and Validation sets

Training performance comparison:

	Gradient Boosting tuned with oversampled data	AdaBoost classifier tuned with oversampled data	Random forest tuned with undersampled data	XGBoost tuned with oversampled data
Accuracy	0.993	0.992	0.961	0.978
Recall	0.992	0.988	0.933	1.000
Precision	0.994	0.995	0.989	0.959
F1	0.993	0.992	0.960	0.979

Validation performance comparison:

	Gradient Boosting tuned with oversampled data	AdaBoost classifier tuned with oversampled data	Random forest tuned with undersampled data	XGBoost tuned with oversampled data
Accuracy	0.969	0.979	0.938	0.938
Recall	0.856	0.853	0.885	0.885
Precision	0.678	0.790	0.468	0.470
F1	0.757	0.820	0.612	0.614

Model Performance Summary continued..

- **Summary of performance metrics for training and validation data for tuned models continued**
  - The validation Recall score for all 4 turned model are high. However the **Precision** for Random Forest and XGBoost is low on validation data.
  - I chose **The tuned Adaboost model** as our final model for the prediction since it has a validation Recall score 0.85, Accuracy 0.97, Precision 0.79 and a F1 score 0.82 on validation data which are the most comparable with the Training performance
- **Model performances and choice of final model**
  - The Validation Recall for Test set after tuning and finalizing the **Adaboost model** is **0.84** with Accuracy **0.97** and Precision **0.78** which is **very comparable** with with the results on test sets build using Pipeline **0.85, 0.97, 0.77**
  - The performance of Tuned Adaboost model on Test set.

Test performance:

	Accuracy	Recall	Precision	F1
0	0.978	0.844	0.788	0.815

Productionize and test the final model using pipelines

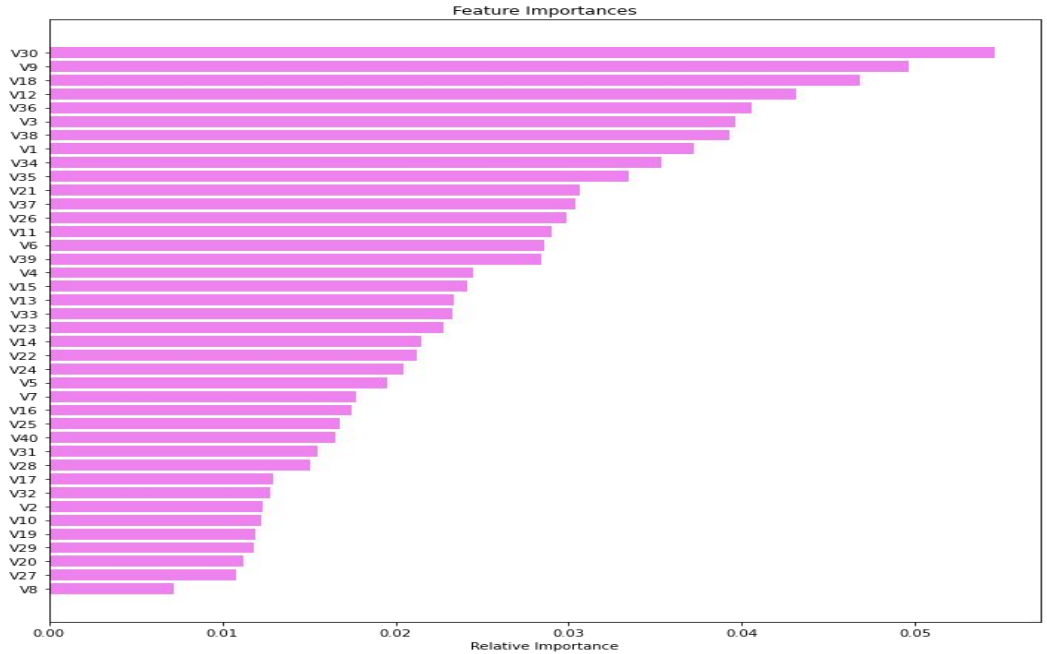
- **Steps taken to create a pipeline for the final model**
  - No need to use column transformer since the data has only one Data Type
  - Separated “Target” and other variables from both Train and Test data sets
  - Treated missing values of the Train set with “median” using imputer.
  - The missing values in the Test set gets imputed inside the pipeline.
  - The chosen best model (Adaboost) is built on the oversampled data.
  - Fit the model on the oversampled data
  - Checked the performance on the Test set

- **Summary of the performance of the model built with pipeline on test dataset**
  - The Recall score is 0.85 compare to the score of the final model performing on unseen data 0.84 using Tuned Adaboost

	Accuracy	Recall	Precision	F1
0	0.978	0.851	0.774	0.811

Productionize and test the final model using pipelines continued..

- **Summary of most important factors used by the model built with pipeline for prediction**
  - The data is given in ciphered version, V30,V9,V18,V12 fields have highest impact on Failure.



Data Preprocessing continued..

- **Model Building**
  - Predictions made by the classification model
    - True positives (TP) are failures correctly predicted by the model.
    - False negatives (FN) are real failures in a generator where there is no detection by model.
    - False positives (FP) are failure detections in a generator where there is no failure.
  - We want Recall to be maximized as greater the Recall (FN), the higher the chances of minimizing false negatives because if a model predicts that a machine will have no failure when there will be a failure, it will increase the maintenance cost.
  - Use Recall as a scorer in cross-validation and hyperparameter tuning
  - Split the Train set into 5 folds for the Cross-Validation performance
  - Built 6 models on original, Oversampled and Undersampled data
    - “Logistic regression”,”Bagging”, “Adaboost”, “Random forest”, “GBM”, “Xgboost”
  - Checked Crossed-Validation on training data and validation performance on validation set.

Model Performance Summary (original data)

- **Summary of performance metrics for training and validation data for comparison for original data.**
  - XGboost, has the highest CV performance score ~ 0.74 followed by Random Forest, Bagging, GBM, Adaboost.
  - Logistic regression has the lowest ~0.5 score.

Original Data

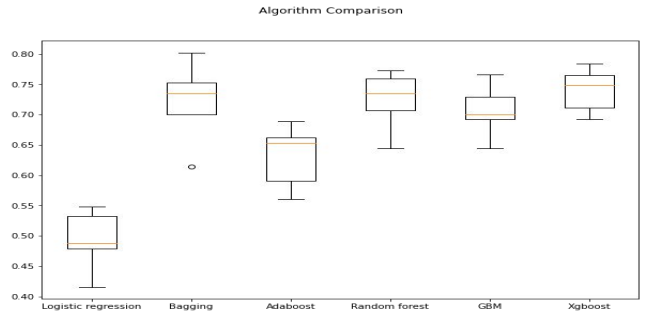
```
Cross-Validation performance on training dataset:

Logistic regression: 0.4927566553639709
Bagging: 0.7210807301060529
Adaboost: 0.6309140754635308
Random forest: 0.7235192266070268
GBM: 0.7066661857008874
Xgboost: 0.7403217661063415

Validation Performance:

Logistic regression: 0.48201438848920863
Bagging: 0.7302158273381295
Adaboost: 0.6762589928057554
Random forest: 0.7266187050359713
GBM: 0.7230215827338129
Xgboost: 0.762589928057554
```

Models Recall Scores on Training set



- **Comments on the model performance**
  - From the chart, Bagging gives the highest cross-validation recall followed by XGboost and Random forest keeping upper whisker in focus.
  - Will Tune XGBoost,Random Forest, Bagging, GBM, Adaboost after checking the scores on oversampled and undersampled data

Model Performance Summary (oversampled data)

- Chose to use oversampling since the dataset suggests an imbalanced class distribution in the “Target” variable. Hance we matched the count of class 1 with class 0.
- **Summary of performance metrics for training and validation data for comparison for oversampled data.**
  - Random Forest has the highest CV performance score 0.98 followed by Bagging, GBM, XGBoost and Adaboost
  - Logistic Regression has the lowest score for the oversampled data

Oversampled Data

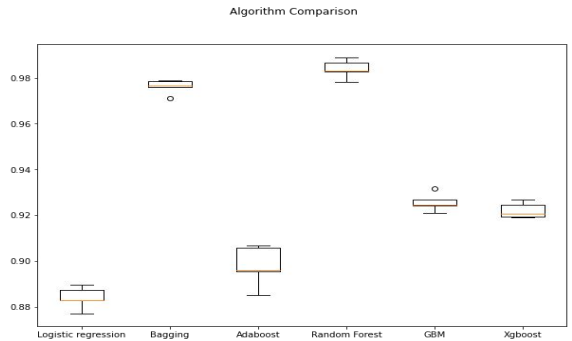
```
Cross-Validation performance on training dataset:

Logistic regression: 0.883963699328486
Bagging: 0.9762141471581656
Adaboost: 0.8978689011775473
Random Forest: 0.9839075260047615
GBM: 0.9256068151319724
Xgboost: 0.922148207398388

Validation Performance:

Logistic regression: 0.8489208633093526
Bagging: 0.8345323741007195
Adaboost: 0.8561151079136691
Random Forest: 0.8489208633093526
GBM: 0.8776978417266187
Xgboost: 0.8741007194244604
```

Models Recall Scores on Training set



- **Comments on the model performance**
  - We can see from the chart, Random forest and Bagging has the highest cross validation Recall
  - Adaboost CV score range is less compared to Bagging, Random Forest, GBM and XGBoost

Model Performance Summary (undersampled data)

- Chose to use oversampling since the dataset suggests an imbalanced class distribution in the “Target” variable. Hance we matched the count of class 0 with class 1.
- **Summary of performance metrics for training and validation data for comparison for undersampled data.**
  - Random Forest and Xgboost has the highest CV performance score 0.90 followed by GBM, Bagging, and Adaboost
  - Logistic Regression has equally good score: 0.87 on undersampled data

Undersampled Data

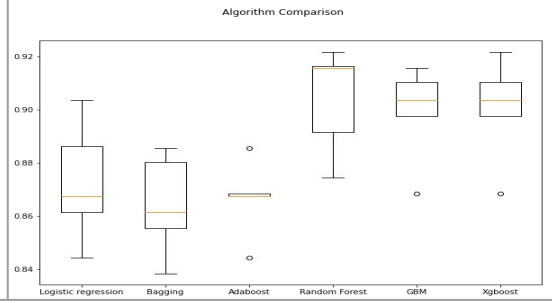
```
Cross-Validation performance on training dataset:

Logistic regression: 0.8726138085275232
Bagging: 0.8641945025611427
Adaboost: 0.8666113556020489
Random Forest: 0.9038669648654498
GBM: 0.8990621167303946
Xgboost: 0.9002669360075031

Validation Performance:

Logistic regression: 0.8525179856115108
Bagging: 0.8705035971223022
Adaboost: 0.8489208633093526
Random Forest: 0.8920863309352518
GBM: 0.8884892086330936
Xgboost: 0.8884892086330936
```

Models Recall Scores on Training set



- **Comments on the model performance**
  - From the chart, Adaboost seems to have the smallest variation in the scores
  - Random Forest has one of the heist score with a large range Vs. XGboost