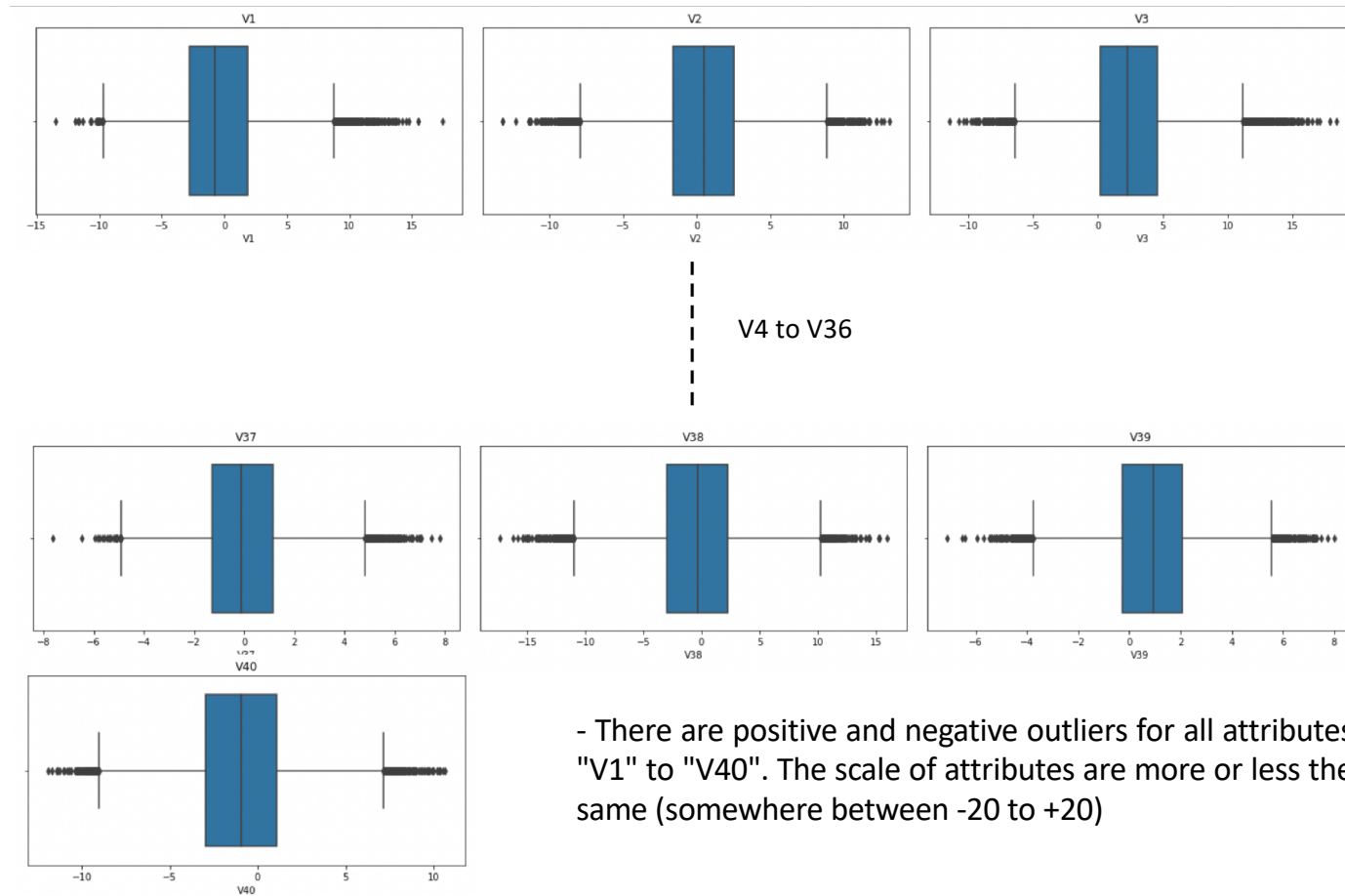


Maintenance Cost Minimization of Machinery for Wind Energy Production Using Machine Learning

- Predictive maintenance – predicting failure patterns for wind energy production machinery to replace components before failure bringing down cost of operations
- ML Predictive Model – Predictions are based off of data collected by sensors fitted to machinery (like temperature, humidity, wind speed etc.) & of parts of wind turbine (gearbox, blades etc.)
- Maintenance cost = (Failures correctly predicted by model)*(Repair cost of \$15K) +
(Failures not predicted by model)*(Replacement cost of \$40K) +
(Failures incorrectly predicted by model)*(Inspection cost of \$5K)
- Data collected consists of about 40,000 rows for ML model training & validation and 10,000 rows for model testing. It has approximately 40 different (encrypted/ confidential) attributes of information on parts & those collected from sensors and 1 target attribute with failure / no-failure status information

Univariate Analysis

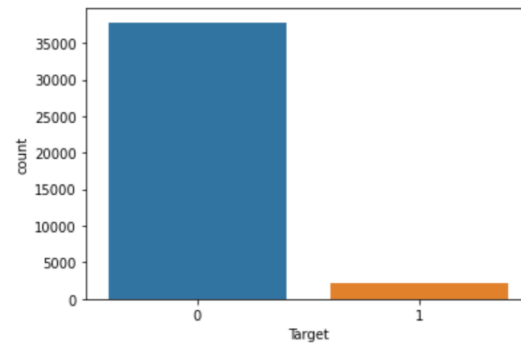
- Boxplot analysis of all 40 attributes (V1 to V40) & target attribute was performed to assess scale and presence of outliers



- There are positive and negative outliers for all attributes "V1" to "V40". The scale of attributes are more or less the same (somewhere between -20 to +20)

Univariate Analysis Continued & Data Pre-processing

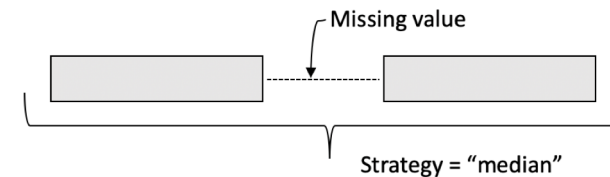
- "Target" class is imbalanced with 37813 or 94.53% "No failures (i.e., 0)" and 2187 or 5.47% "Failures (i.e., 1)"



- Training set was split 70:30 for model training (fitting) & model validation (further optimization & tuning). Target class imbalance was maintained across training & validation splits



- Median of the attribute was used to impute any missing values in the attributes (i.e., for "V1" and "V2" which had 46 & 39 missing values)
 - This was performed separately on training set and validation set to prevent any data leak!



* Performed twice on training & validation set independently

Model Evaluation Criteria & Model Building

- Creation of a customized metric:

(Minimum Cost)/(Cost Associated with the Machine Learning model)

Higher the value of the customized metric, better the performance of the model

- 7 different machine learning algorithms were fit on original training dataset

- Logistic Regression
- Decision Tree
- Random Forest
- Bagging Classifier
- Boosting Classifiers (AdaBoost, Gradient Boost, XGBoost)

*** 5 fold cross validation was performed to calculate an average customized metric score on training data set**

- Class imbalance was handled by

- Synthetic Minority Oversampling Technique (SMOTE)
- Random Under sampler

& the 7 machine learning algorithms were fit again

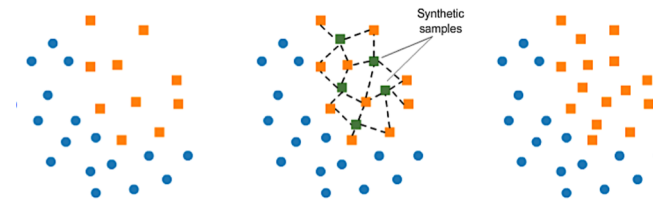
- **5 fold cross validation was performed to calculate an average customized metric score on training dataset**

- Model performance was evaluated on validation data set to assess if model generalizes well or is prone to overfitting/underfitting

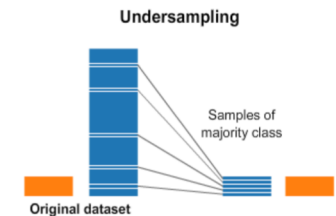
		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

Customized Metric:

$$((TP + FN) * 15) / (TP * 15 + FP * 5 + FN * 40)$$






SMOTE – adding synthetic minority class data points




Random under sampler of majority class

Model Performance & Selection

	Cross-Validation Training dataset Performance (customized metric)	Validation dataset Performance (customized metric)	
Original data set	<ul style="list-style-type: none"> Logistic Regression: 0.53 dtree: 0.65 Random forest: 0.71 Bagging: 0.68 Adaboost: 0.60 GBM: 0.68 Xgboost: 0.77 	<ul style="list-style-type: none"> Logistic Regression: 0.52 dtree: 0.66 Random forest: 0.72 Bagging: 0.68 Adaboost: 0.59 GBM: 0.67 Xgboost: 0.77 	 Generalized performance
SMOTE	<ul style="list-style-type: none"> Logistic Regression: 0.79 dtree: 0.94 Random forest: 0.97 Bagging: 0.96 Adaboost: 0.83 GBM: 0.87 Xgboost: 0.97 	<ul style="list-style-type: none"> Logistic Regression: 0.50 dtree: 0.64 Random forest: 0.80 Bagging: 0.76 Adaboost: 0.57 GBM: 0.73 Xgboost: 0.80 	 Overfitting
Random Under sampler	<ul style="list-style-type: none"> Logistic Regression: 0.77 dtree: 0.76 Random forest: 0.84 Bagging: 0.81 Adaboost: 0.79 GBM: 0.82 Xgboost: 0.84 	<ul style="list-style-type: none"> Logistic Regression: 0.50 dtree: 0.47 Random forest: 0.73 Bagging: 0.67 Adaboost: 0.55 GBM: 0.68 Xgboost: 0.74 	 Overfitting


- Models built on original dataset have given generalized performance on 5-fold cross-validation training and validation datasets unlike models built on oversampled or undersampled data sets which may be prone to overfitting & slight underfitting respectively
- Mean cross validation scores on training sets are highest with XGBoost, Random Forest & Bagging Classifiers (~77, ~71 and ~68% respectively). These models will be tuned further to try to increase performance

Hyperparameter Tuning (RandomizedSearchCV)

	Hyperparameter Tuning (Best parameters)	Cross-Validation Training dataset Performance (customized metric)	Validation dataset Performance (customized metric)
XGBoost	<ul style="list-style-type: none"> 'subsample': 0.9 'scale_pos_weight': 10 'n_estimators': 250 'learning_rate': 0.1 'gamma': 3 	0.80	0.82 
Random forest	<ul style="list-style-type: none"> 'n_estimators': 250 'min_samples_leaf': 1 'max_samples': 0.5 'max_features': 'sqrt' 	0.69	0.69
Bagging	<ul style="list-style-type: none"> 'n_estimators': 50 'max_samples': 0.9 'max_features': 0.8 	0.71	0.70

Validation performance comparison:

	XGBoost Tuned with Random search	Random forest Tuned with Random search	Bagging Tuned with Random Search
Accuracy	0.991	0.985	0.985
Recall	0.877	0.741	0.745
Precision	0.962	0.988	0.978
F1	0.917	0.847	0.846
Minimum_Vs_Model_cost	0.821	0.697	0.699

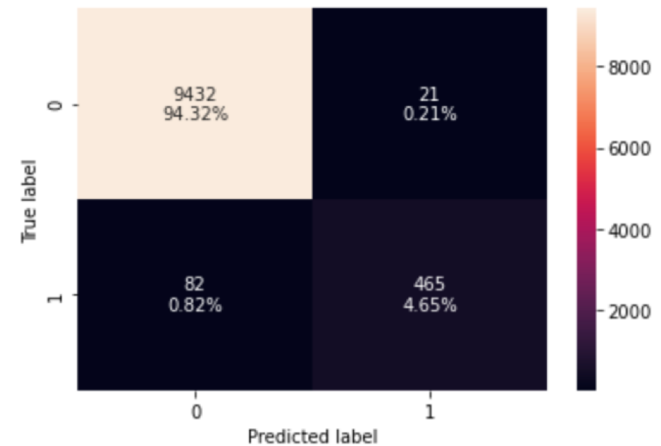
 **Final model**

- The XGBoost Tuned model with Random Search is giving the highest performance score (Minimum_Vs_Model_cost) of 0.821 on the Validation Set (The average cross validation Training performance score (Minimum_Vs_Model_cost) with this model is 0.80 indicating model may generalize well)
- Accuracy, Precision and F1 scores are likewise high

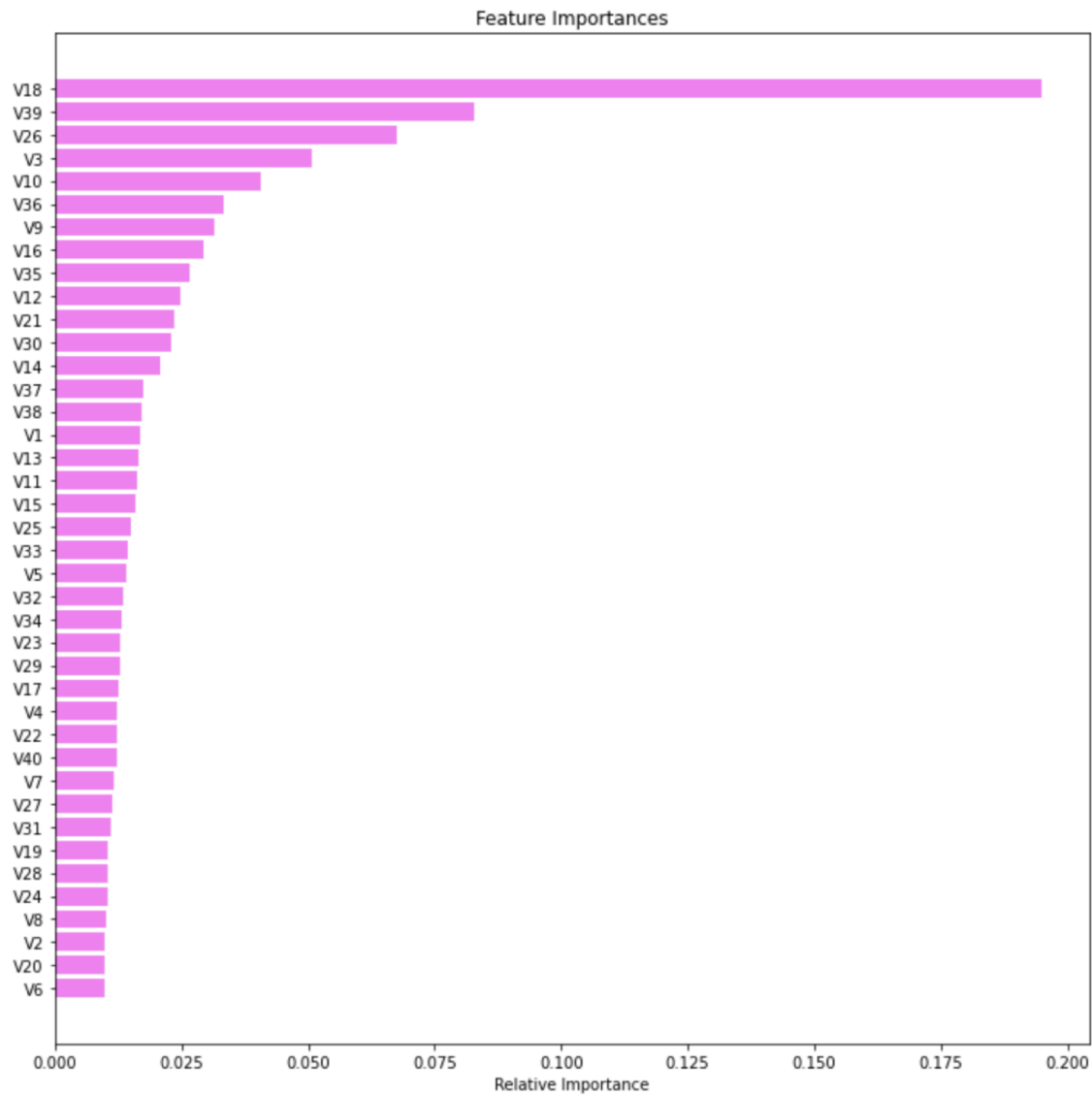
Final Model – Test Performance

Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0.990	0.850	0.957	0.900	0.792

- The XGBoost tuned model is generalizing well on the test data with a Minimum_Vs_Model_cost of 0.792. The model is able to make predictions resulting in a maintenance cost $\sim (1/0.792 \text{ or } \sim 1.26)$ times the minimum maintenance cost possible
- From the confusion matrix, we can see the % for both False negatives (0.82%) and False positives (0.21%) are very small. The accuracy and precision scores for the model are high (0.95+) and the F1 score & recall are decent (0.9 and 0.85)
- Furthermore, the data pre processing steps & XGBoost tuned learning model were automated by building pipelines (for productionisation)



Minimum_Vs_Model_cost:
 $((TP + FN) * 15) / (TP * 15 + FP * 5 + FN * 40)$



- The top attributes which have the maximum importance for making accurate failure/ no-failure predictions are "V18", "V39", "V26", "V3" & "V10"

Business Insights & Conclusions

- 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 & further optimizing for target class imbalance (having few "failures" and many "no failures" in dataset) as well as finetuning the algorithm performance (hyperparameter and cross validation techniques)
 - A pipeline was additionally built to productionise the final chosen model
- The model is expected to generalize well in terms of predictions & expected to result in a maintenance cost ~1.26 times minimum possible maintenance cost. Having no model in place for predictions could potentially result in costs as high as ~2.67 minimum possible maintenance cost. Hence, productionising the model has a large cost saving advantage
- The main attributes of importance for predicting failures vs. no failures were found to be "V18", "V39", "V26", "V3" & "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