ReneWind Presentation

Business Problem Overview and Solution Approach

Context

Renewable energy sources play an increasingly important role in the global energy mix, as the effort to reduce the environmental impact of energy production increases.

Out of all the renewable energy alternatives, wind energy is one of the most developed technologies worldwide. The U.S Department of Energy has put together a guide to achieving operational efficiency using predictive maintenance practices.

Predictive maintenance uses sensor information and analysis methods to measure and predict degradation and future component capability. The idea behind predictive maintenance is that failure patterns are predictable and if component failure can be predicted accurately and the component is replaced before it fails, the costs of operation and maintenance will be much lower.

The sensors fitted across different machines involved in the process of energy generation collect data related to various environmental factors (temperature, humidity, wind speed, etc.) and additional features related to various parts of the wind turbine (gearbox, tower, blades, break, etc.).

Business Objective

Objective

“ReneWind” is a company working on improving the machinery/processes involved in the production of wind energy using machine learning and has collected data of generator failure of wind turbines using sensors. They have shared a ciphered version of the data, as the data collected through sensors is confidential (the type of data collected varies with companies). Data has 40 predictors, 40000 observations in the training set and 10000 in the test set.

The objective is to build various classification models, tune them and find the best one that will help identify failures so that the generator could be repaired before failing/breaking and the overall maintenance cost of the generators can be brought down.

“1” in the target variables should be considered as “failure” and “0” will represent “No failure”.

Business Objective Analytics

The nature of predictions made by the classification model will translate as follows:

- True positives (TP) are failures correctly predicted by the model.
- False negatives (FN) are real failures in the generator of wind turbine where there is no detection by the model.
- False positives (FP) are failure detections in the generator of the wind turbine where actually there is no failure.

So, the maintenance cost associated with the model would be:

- Maintenance cost = TP*(Repair cost) + FN*(Replacement cost) + FP*(Inspection cost)

where,

- Replacement cost = $40,000
- Repair cost = $15,000
- Inspection cost = $5,000

Here the objective is to reduce the maintenance cost so, we want a metric that could reduce the maintenance cost.

The minimum possible maintenance cost = Actual failures*(Repair cost) = (TP + FN)*(Repair cost)

And the maintenance cost associated with model = TP*(Repair cost) + FN*(Replacement cost) + FP*(Inspection cost)

So, we will try to maximize the ratio of minimum possible maintenance cost and the maintenance cost associated with the model.

The value of this ratio will lie between 0 and 1, the ratio will be 1 only when the maintenance cost associated with the model will be equal to the minimum possible maintenance cost.

Data Description

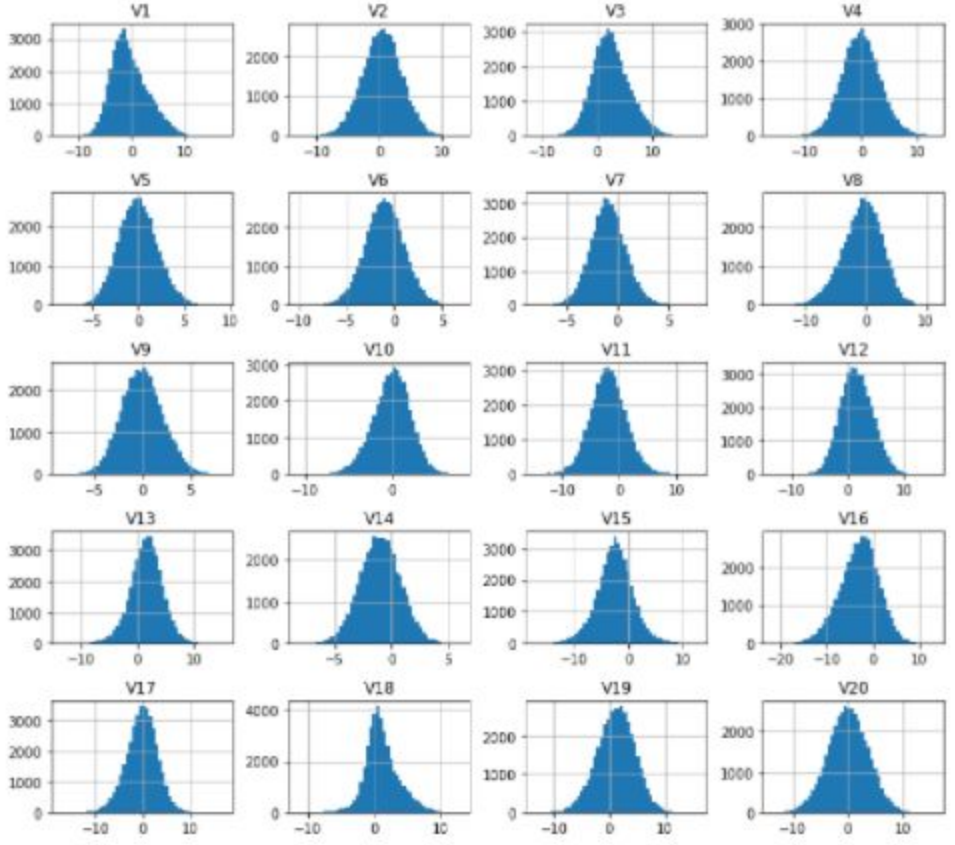
- Two data sets were provided, one with 40,000 data points will be used as training/validation data while the remaining with 10,000 data points will be used separately as the test data set in order to prevent data leakage from modeling into production models.
- Training Data is comprised of 40 predictors and each has 1000 data points.
- 40,000 data points with one Target column for whether or not the unit failed.
- All of the data points are of object type float, with the exception being the Target column, which is of type integer in the 0/1 format.
- V1 column has 0.12% missing values out of the total observations.
- V2 column has 0.10% missing values out of the total observations.
- Values range from negative teens to positive teens
- Distributions of the data points and the similarity of standard deviations, mean and median values indicate bell curve shaped data with the differences in predictors being how positive or negative the curve extends to and also the count height.

Univariate Analysis: Histogram Distributions (Column 1-20)

Initial analysis of the data showed a normal bell curve tendency for the data.

Histogram presentations reiterate what was seen during data preparation, that is the similarity of all the data which makes sense as the sensors on the equipment would be of similar make.

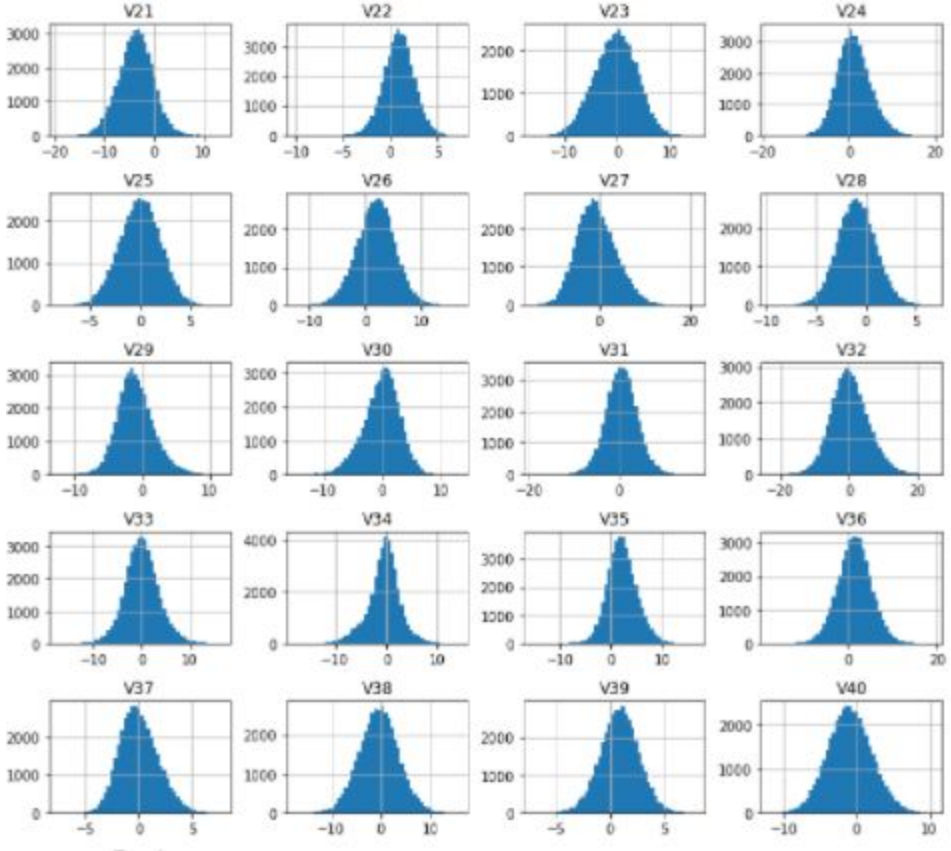
Analysis of the first 20 predictors show counts between 2000 to slightly over 3000 for the predictors with ranges of values from -10 to +10, with most predictors having outliers.



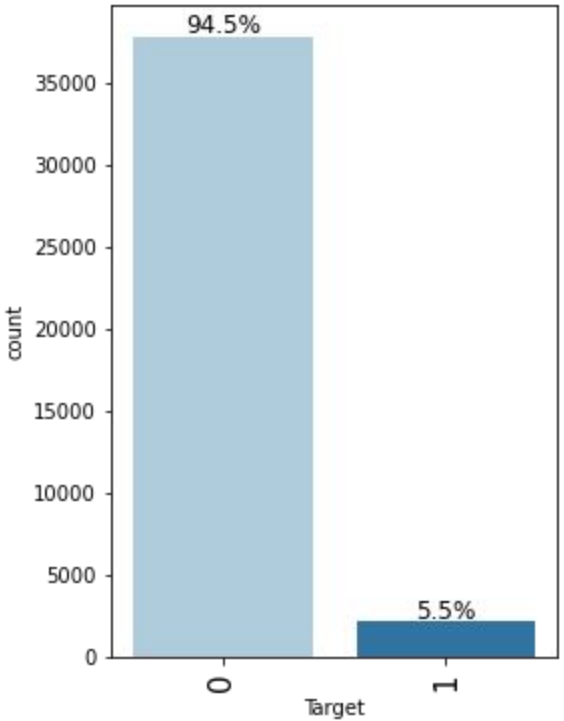
Univariate Analysis: Histogram Distributions (Column 21-40)

Analysis of predictor columns 21 -40 show distributions similar to the first 20 columns.

Counts vary from 2000 to over 3000 with ranges from negative teens to positive teens and all predictors displaying bell curve shapes with outliers on both ends.



Univariate Analysis: Target Column



Target column shows an equipment fail rate of 5.5%

The model must attempt to predict the failure of the equipment prior to the system going to failure as replacing costs more than maintenance.

Data Pre-Processing:

- Missing values (those with N/A) were treated by applying the median column value to the missing values.

- The set was divided into training and validation sets in order to maintain data integrity and prevent data leakage.

```
Training
0    0.945321
1    0.054679
Name: Target, dtype: float64
**********************************
Validation
0    0.945333
1    0.054667
Name: Target, dtype: float64
```

Training data set has 28,000 data points while validation has 12,000 data points.

Rate of failure (1) and rates of not failing (0) were consistent between both groups.

Modeling Criteria:

Model evaluation criterion

3 types of cost are associated with the provided problem

1. Replacement cost - False Negatives - Predicting no failure, while there will be a failure
2. Inspection cost - False Positives - Predicting failure, while there is no failure
3. Repair cost - True Positives - Predicting failure correctly

How to reduce the overall cost?

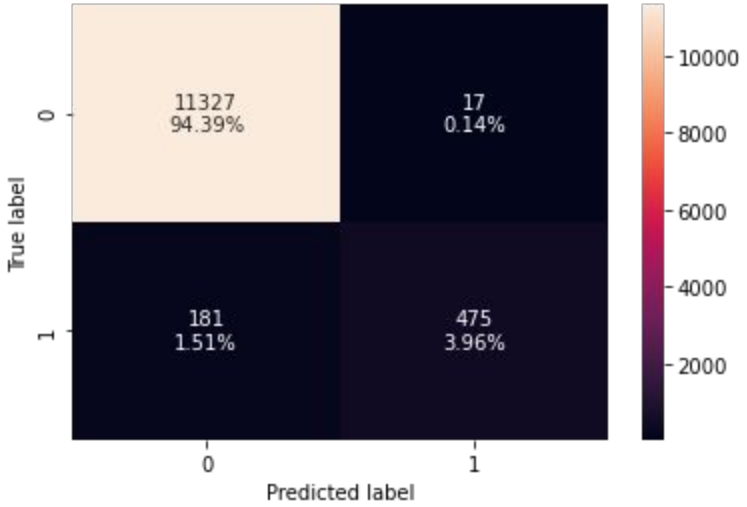
- We need to create a customized metric, that can help to bring down the overall cost.
- The cost associated with any model = TP * 15000 + FP * 5000 + FN * 40000
- And the minimum possible cost will be when, the model will be able to identify all failures, in that case, the cost will be (TP + FN) * 15000
- So, we will try to maximize `Minimum cost/Cost associated with model`

Modeling Plan:

-The training data will have multiple models built on it and then the data will have oversampling techniques (SMOTE) applied and the same models applied to that data. The training data will also be undersampled and performance of the models will be tested on that as well to compare to normal data and oversampled data.
-Due to the nature of the data, in that the fail is such a low percentage and total out of 40,000 data points the oversampling of data may be the best method once comparisons are made.
- Models to be used:
    - Bagging Classifier
    - Decision Tree
    - Random Forest
    - Gradient Boost
    - AdaBoost
    - XGB Boost

Bagging Classifier on Original Data:

- Bagging classifier on original data shows good metrics on training, but the min vs model cost is much lower than our goal of 0.78
- This model is overfitting the trained data as the recall drops a lot from training to validation



Training performance:

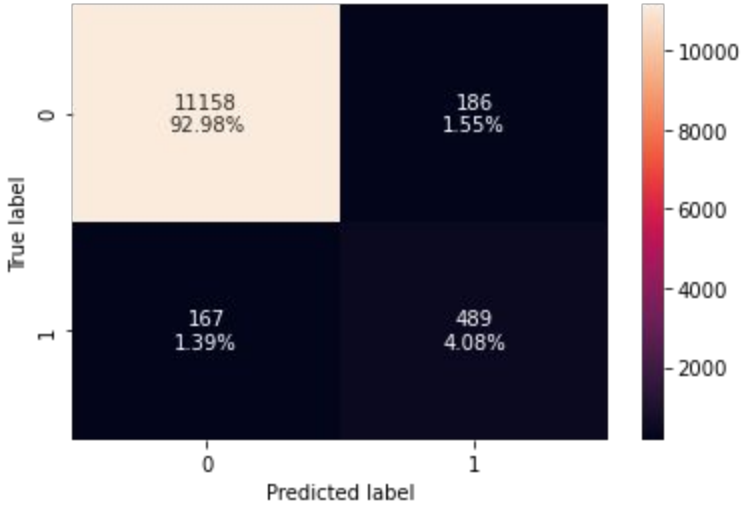
	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.997679	0.957544	1.0	0.978312	0.933916

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.9835	0.724085	0.965447	0.827526	0.680969

Decision Tree on Original Data:

- Decision Tree model on original data shows generalized performance on the validation set and a decrease in recall and precision scores from training to validation.
- The min vs model cost is also lower than our goal



Training performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	1.0	1.0	1.0	1.0	1.0

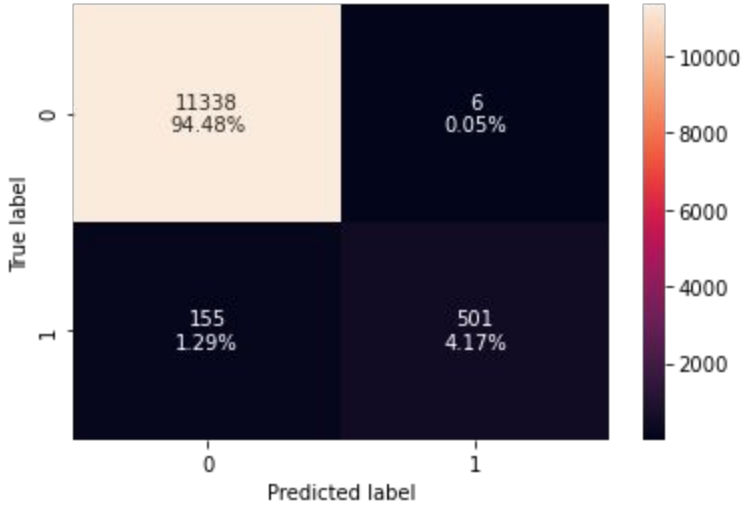
Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.970583	0.745427	0.724444	0.734786	0.658414

Random Forest on Original Data:

- Random Forest modeling on original data shows fairly good results between the training and validation sets.

- Random Forest also has the highest min vs model cost so far at over 0.71



Training performance:

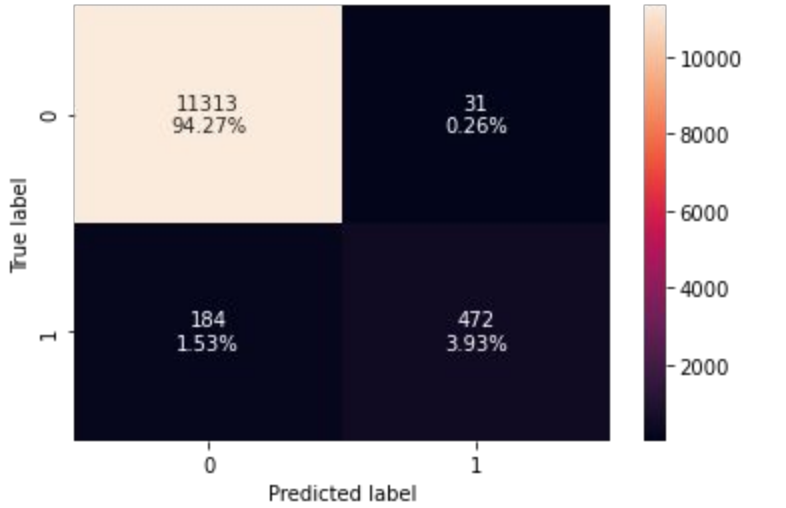
	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.999964	0.999347	1.0	0.999673	0.998913

Testing performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.986583	0.76372	0.988166	0.861565	0.715897

Gradient Boost Method on Original Data:

- Gradient Boost is showing good metrics between the training and validation sets.
- The min vs model cost is lower on both training and validation sets when compared to previous modeling methods.



Training performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.987214	0.782495	0.979558	0.870007	0.73102

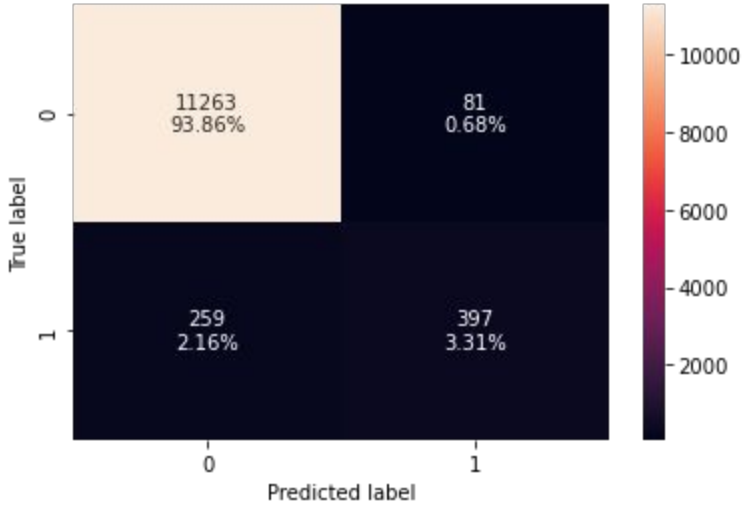
Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.982083	0.719512	0.93837	0.814495	0.674203

AdaBoost Modeling on Original Data:

- AdaBoost modeling has low recall on both training and validation sets as well as very low min vs model cost on both sets as well.

  Confusion matrix still shows same general tendencies like previous models.



Training performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.975714	0.627041	0.898036	0.738462	0.607781

Validation performance:

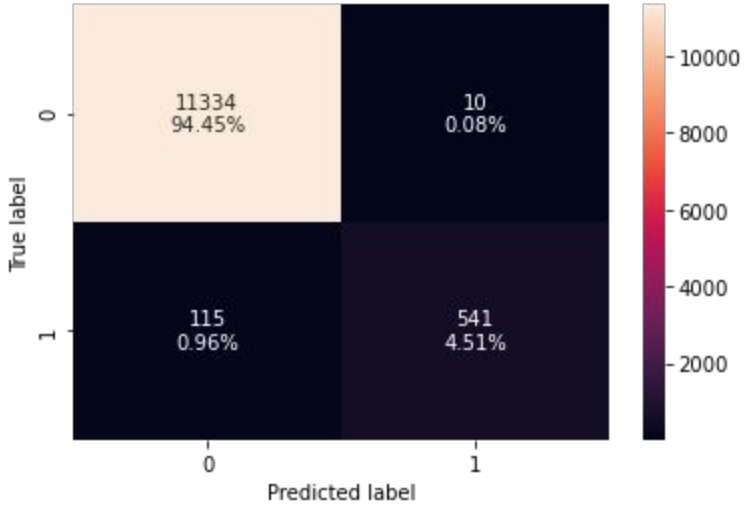
	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.971667	0.605183	0.830544	0.700176	0.588517

XGB Boost on Original Data:

- XGB Boost is showing the best scoring metrics across all the models used.
- Validation data also has very good accuracy, recall and precision.

  Model may be overfitting the training data slightly.

  Min vs model cost is also high being 0.77, which is close to our target of at least 0.78.



Training performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	1.0	1.0	1.0	1.0	1.0

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.989583	0.824695	0.981851	0.896437	0.770858

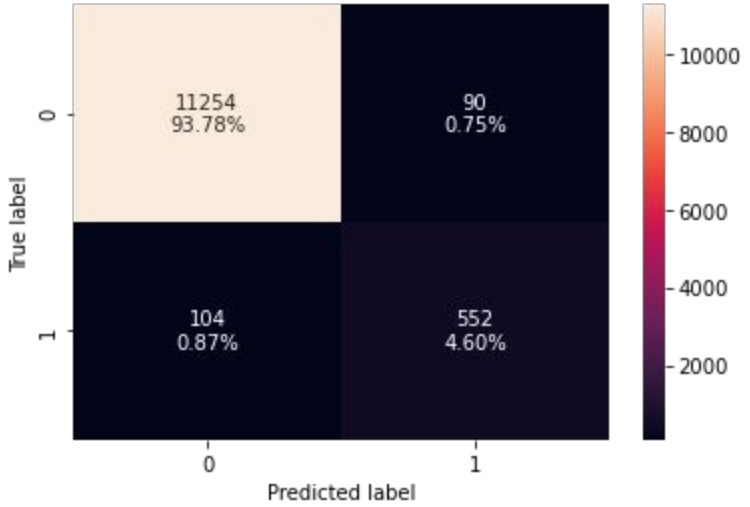
Oversampled Data:

- SMOTE (Synthetic Minority Over Sampling Technique) will be used to oversample the training data in order to attempt to increase representation of the minority group (failure in target column).
- K- Nearest Neighbors value of “5” was used.

```
# Synthetic Minority Over Sampling Technique
sm = SMOTE(sampling_strategy=1, k_neighbors=5, random_state=1)
X_train_over, y_train_over = sm.fit_resample(X_train, y_train)
```

Bagging Classifier on Over Sampled Data:

- Bagging classifier on the over sampled data has improved performance from original data model.
- All scoring metrics are showing very well.
- Oversampling the data has increased the min vs model cost on the validation set.



Training performance:

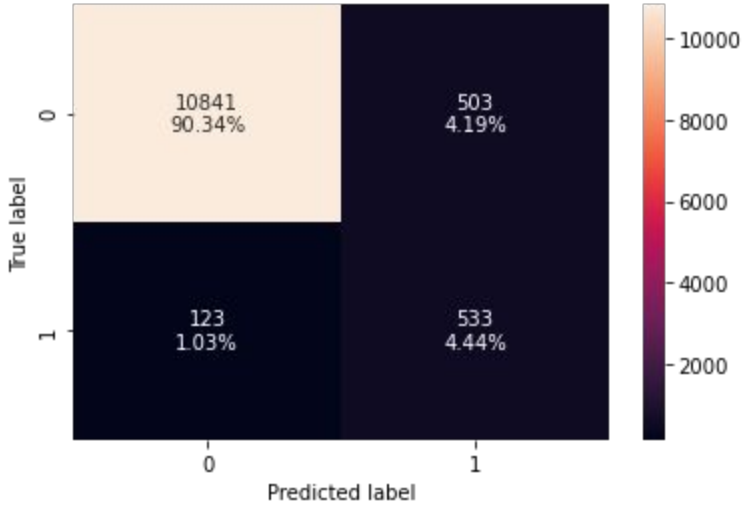
	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.999226	0.998602	0.999849	0.999225	0.997626

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.983833	0.841463	0.859813	0.850539	0.763382

Decision Tree on Over Sampled Data:

- Decision Tree on Over Sampled Data is showing lower metrics than on the original data. Precision has lowered a lot.
- The min vs model cost on the validation data has also lowered



Training performance:

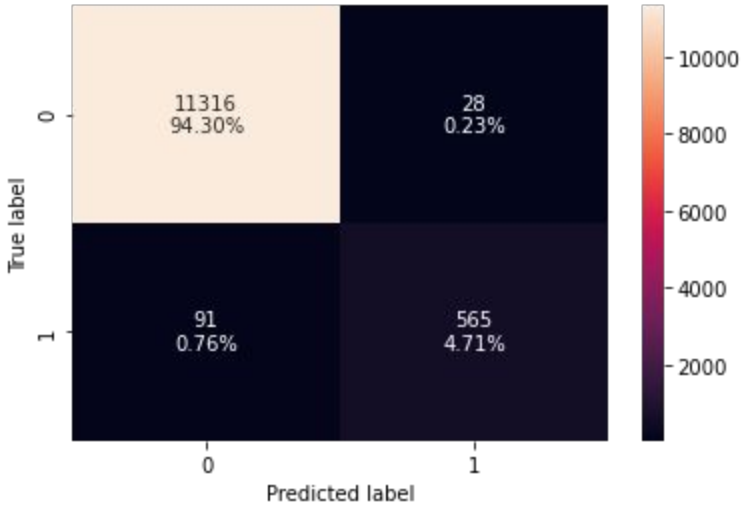
	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	1.0	1.0	1.0	1.0	1.0

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.947833	0.8125	0.514479	0.630024	0.637719

Random Forest on Over Sampled Data:

- Random Forest on the Over Sampled data is showing very good metrics.
- 94.3% correct identification of non failing predictors.
- Min vs model cost on the validation set is also the highest we have seen so far at 0.80.
- Model may be overfitting the training data slightly with the 100% metrics across the board.



Training performance:

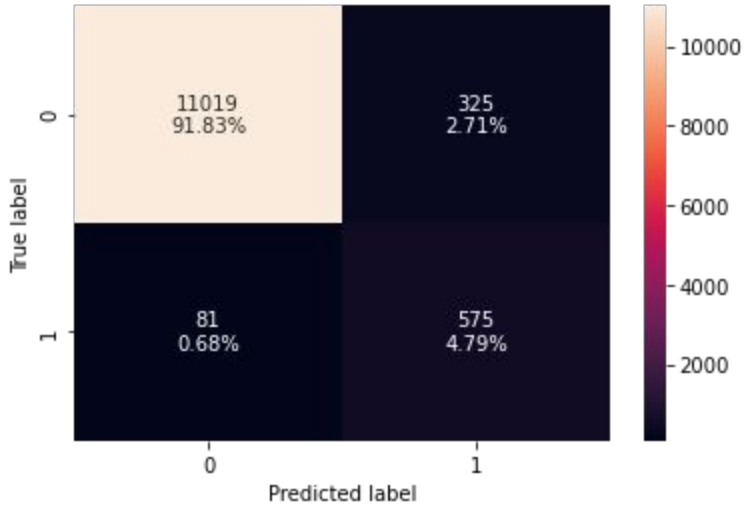
	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	1.0	1.0	1.0	1.0	1.0

Testing performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.990083	0.86128	0.952782	0.904724	0.802938

Gradient Boost on Over Sampled Data:

- Gradient Boost on the Over Sampled data is showing fairly good metrics.
- Confusion matrix still shows over 90% correctly identifying the failing equipment, which has been the trend across all models in this data set.
- Precision has taken a drop as well as F1 score with this modeling method.
- Min vs Model cost scores on the validation data are still good though



Training performance:

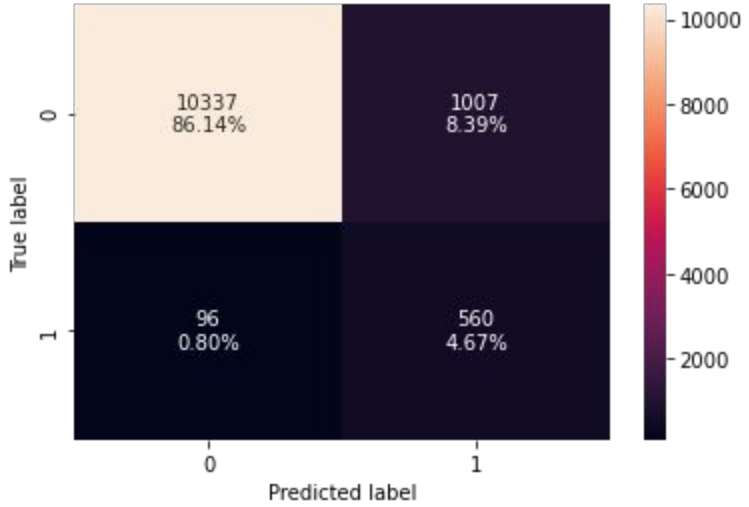
	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.945427	0.917828	0.971449	0.943878	0.872643

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.966167	0.876524	0.638889	0.739075	0.729429

AdaBoost on Over Sampled Data:

- AdaBoost on Over Sampled data has dropped the correct identification to 86%, down from over 90% on the original data.
- The validation precision, F1 and min vs model cost are all also much lower than on original data, and much lower than desired.



Training performance:

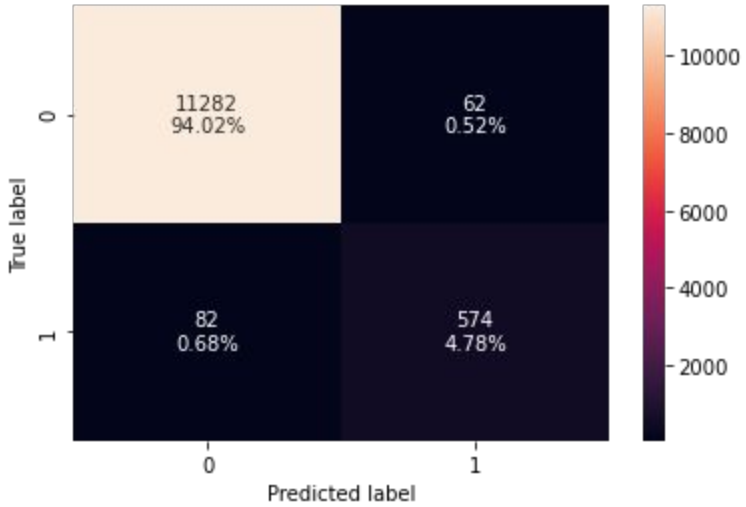
	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.907326	0.897654	0.91536	0.90642	0.834554

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.908083	0.853659	0.357371	0.503824	0.569609

XGB Boost on Over Sampled Data:

- XGB boost on the Over Sampled data shows very good performance.
- Performance on the training set has dropped from 100% to 99%.
- Validation scores are still very good and the min vs model cost of 0.81 is also very good.
- Recall and min vs model cost have increased from the XGB boost on original data.



Training performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.99915	0.998867	0.999433	0.99915	0.997926

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.988	0.875	0.902516	0.888545	0.806557

Under Sampling with Random Under Sampler:

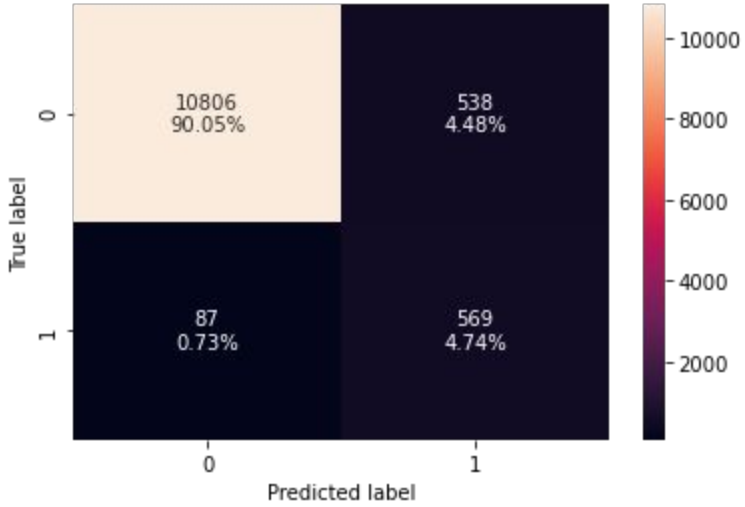
- Data will be modified using Random Under Sampler in order to bring down the majority classes to even out the distribution.
-

```
# Random undersampler for under sampling the data
rus = RandomUnderSampler(random_state=1, sampling_strategy=1)
X_train_un, y_train_un = rus.fit_resample(X_train, y_train)
```

Bagging Classifier on Under Sampled Data:

- Bagging Classifier on the Under Sampled data has lowered the validation performance, specifically the precision and F1 scores.

- The min vs model cost has also lowered from both the original data and the Over Sampled data.



Training performance:

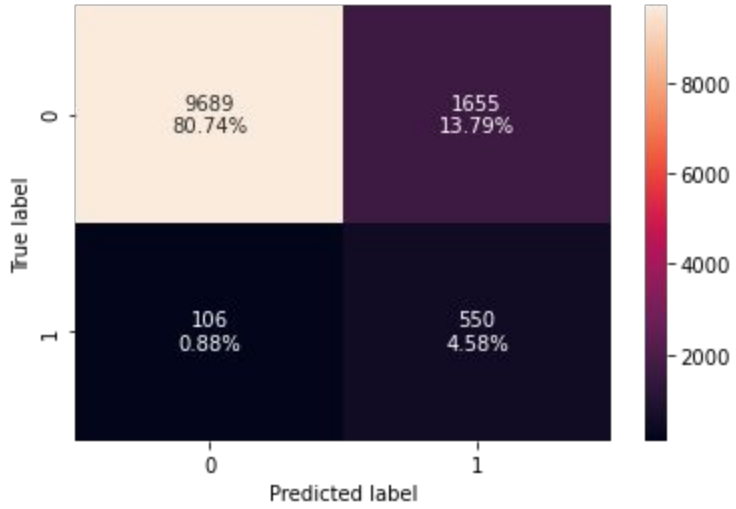
	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.987263	0.977139	0.997333	0.987133	0.96249

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.947917	0.867378	0.514002	0.645491	0.66916

Decision Tree on Under Sampled Data:

- Decision Tree with Under Sampled data has the lowest metrics so far.
- This model has the lowest True Positive ID rate of 80.74%.
- Precision and F1 are now extremely low and even the min vs model cost score is showing the lowest so far at 0.47.



Training performance:

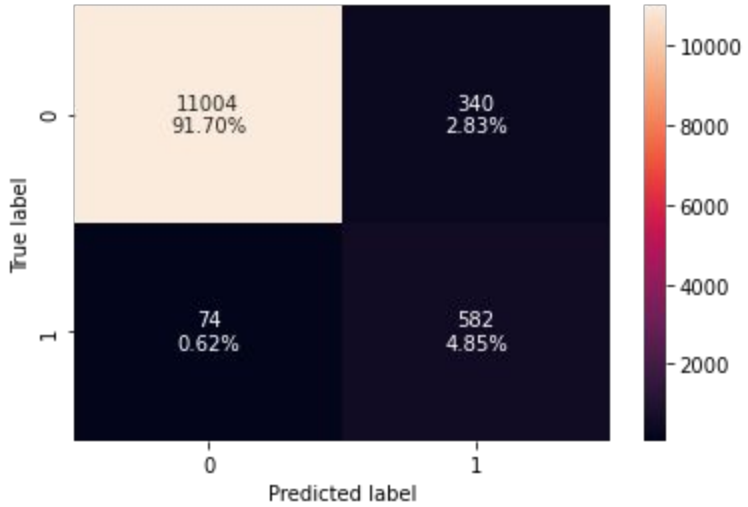
	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	1.0	1.0	1.0	1.0	1.0

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.85325	0.838415	0.249433	0.384481	0.473874

Random Forest on Under Sampled Data:

- Random Forest on Under Sampled Data is still showing good performance.
- Similar to the other two Random Forest models, this one is one of the highest across the validation testing metrics.
- Min vs Model cost of 0.73 is also one of the highest so far across all models



Training performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	1.0	1.0	1.0	1.0	1.0

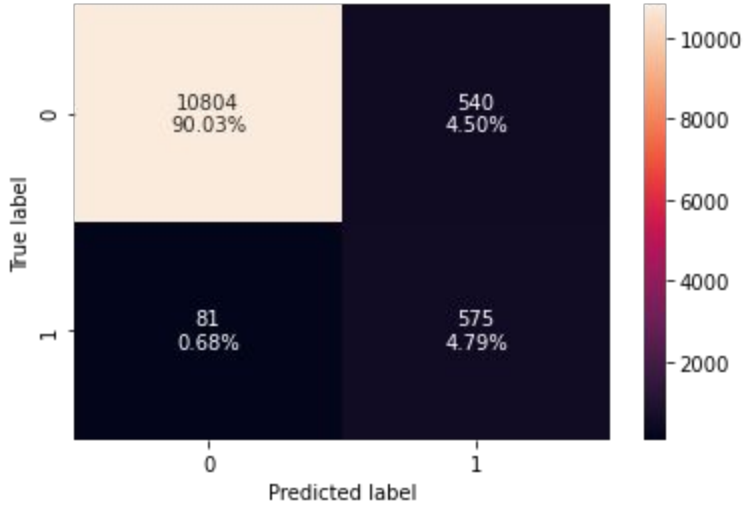
Testing performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.9655	0.887195	0.631236	0.737643	0.734877

Gradient Boost on Under Sampled Data:

- Gradient Boost on Under Sampled data has good performance metrics and those metrics do not drop off between training and validation data sets.

- However, both recall and the metric of interest min. Vs model cost are both very low.



Training performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.987214	0.782495	0.979558	0.870007	0.73102

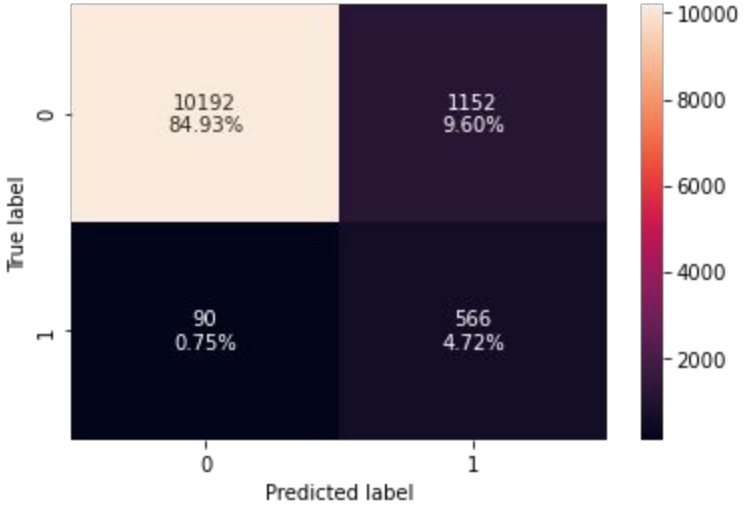
Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.982083	0.719512	0.93837	0.814495	0.674203

AdaBoost on Under Sampled Data:

- AdaBoost shows extreme overfitting with the very good training data scores versus the very low precision and F1 scores on the validation data.

- Validation data has a min vs model cost of only 0.55, one of the lowest across all models.



Training performance:

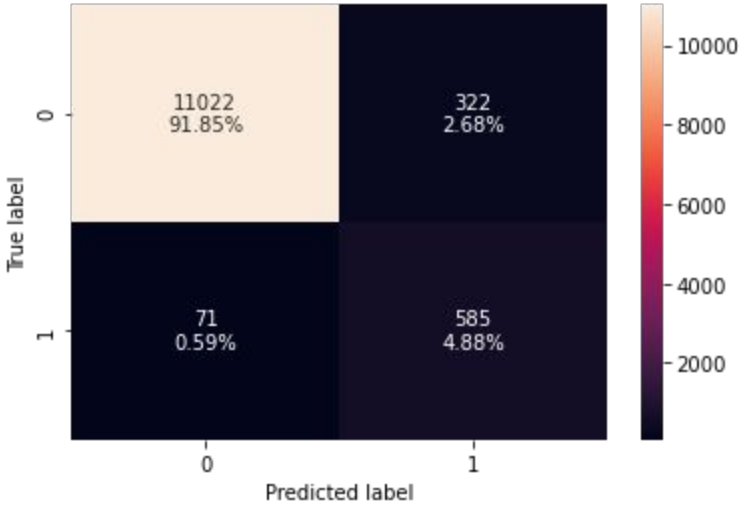
	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.906597	0.881777	0.927835	0.90422	0.819739

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.8965	0.862805	0.329453	0.476832	0.551261

XGB Boost on Under Sampled Data:

- XGB boost on Under Sampled data shows decent performance across all metrics.
- The model overfits on the training data and that affects precision and F1 score on the validation data.
- The model does have good validation data accuracy and recall.
- The min vs model cost is slightly lower than our target of 0.74



Training performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	1.0	1.0	1.0	1.0	1.0

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.96725	0.891768	0.644983	0.74856	0.744045

Modeling Observations:

- The best prediction models are coming from models on the Over Sampled data set as this creates less of a disparity between the majority and the minority variables.
- The strongest models showing good metrics between training and validation data are Bagging Classifier, Random Forest, and XGB Boost using the Over Sampled Data.
- These three models have good accuracy, predictability shown by confusion matrix, recall (to reduce false negatives), as well as very good minimum vs model cost ratio of clost to 0.78, which is the necessary low end to produce a good model that will offer insight and also the best cost-benefit.
- These three models using Over Sampled Data will have Hyperparameter tuning applied to see which of the three offers the best predictive model.

XGB Boost tuning on Over Sampled Data:

```
Best parameters are {'subsample': 0.8, 'scale_pos_weight': 10, 'n_estimators': 250, 'learning_rate': 0.1, 'gamma': 5} with CV s
core=0.9956553162608432:
```

Model:

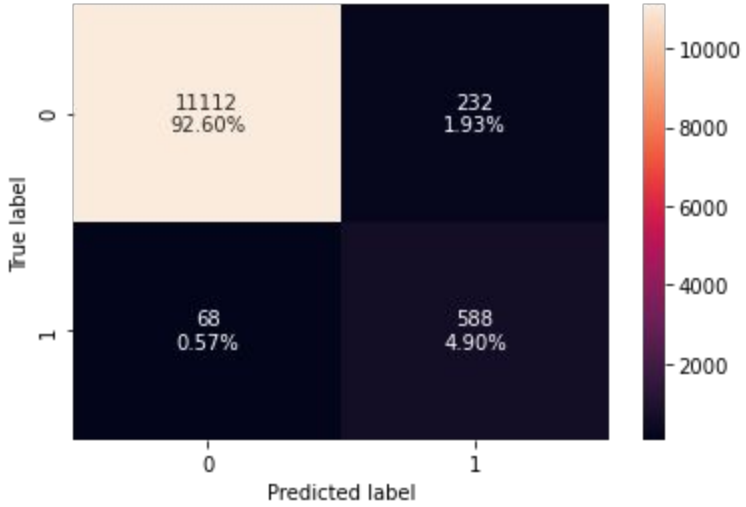
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, eval_metric='logloss',
              gamma=5, gpu_id=-1, importance_type='gain',
              interaction_constraints='', learning_rate=0.1, max_delta_step=0,
              max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=250, n_jobs=8,
              num_parallel_tree=1, random_state=1, reg_alpha=0, reg_lambda=1,
              scale_pos_weight=10, subsample=0.8, tree_method='exact',
              validate_parameters=1, verbosity=None)
```

Training performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.991509	1.0	0.983301	0.99158	0.994371

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.975	0.896341	0.717073	0.796748	0.774803



- XGB Boost with tuned hyperparameters on the Over Sampled Data has very strong results.
- Good Validation metrics as well as a min vs model cost of 0.77

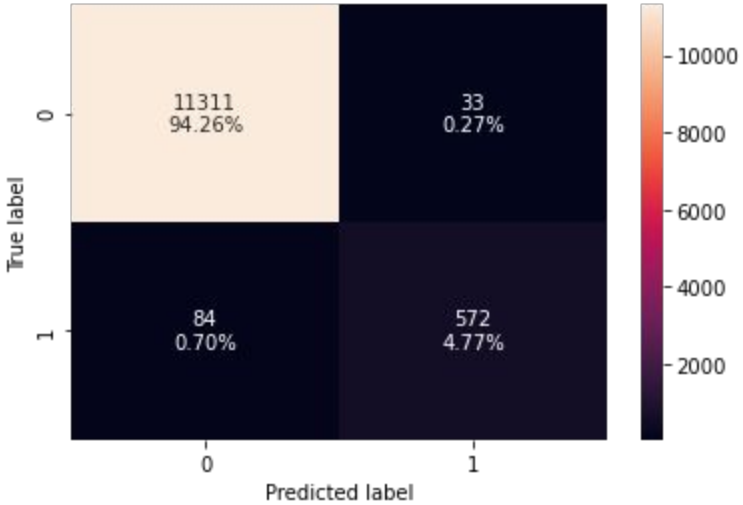
Random Forest tuning on Over Sampled Data:

```
Best parameters are {'n_estimators': 250, 'min_samples_leaf': 1, 'max_samples': 0.5000000000000001, 'max_features': 'log2'} wit
h CV score=0.9752163705533488:
```

Model:

```
RandomForestClassifier(max_features='log2', max_samples=0.5000000000000001,
                       n_estimators=250, random_state=1)
```

```
Training performance:
    Accuracy    Recall  Precision        F1  Minimum_Vs_Model_cost
0  0.998999  0.998375   0.999622  0.998998               0.997175
Validation performance:
    Accuracy    Recall  Precision        F1  Minimum_Vs_Model_cost
0   0.99025  0.871951   0.945455  0.907216               0.812887
```



- Random Forest with tuned hyperparameters on the Over Sampled Data has the highest metrics so far on validation data, including a min vs model cost of 0.81.
- Model is correctly ID'ing 94.26% of the equipment.

Bagging Classifier tuned on Over Sampled Data:

```
Best parameters are {'n_estimators': 50, 'max_samples': 0.9, 'max_features': 0.8} with CV score=0.9794854256832217:
```

Model:

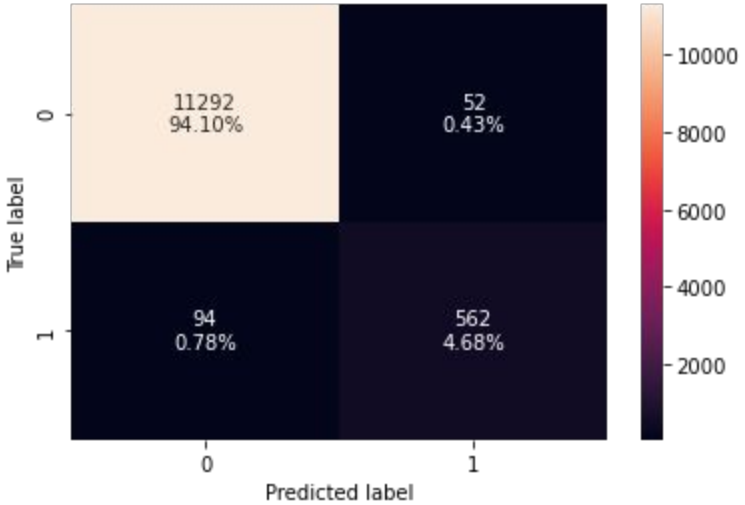
```
BaggingClassifier(max_features=0.8, max_samples=0.9, n_estimators=50,
                  random_state=1)
```

Training performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.999981	0.999962	1.0	0.999981	0.999937

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.987833	0.856707	0.915309	0.885039	0.790361



- Bagging Classifier tuned on Over Sampled Data is showing very good model metrics, with min vs model cost of 0.79, which is above our threshold of 0.78.

Model Performance Comparison:

Training performance comparison:

	Bagging Tuned with Random search	Random Forest Tuned with Random search	Xgboost Tuned with Random Search
Accuracy	0.999981	0.998999	0.991509
Recall	0.999962	0.998375	1.000000
Precision	1.000000	0.999622	0.963301
F1	0.999981	0.998998	0.991580
Minimum_Vs_Model_cost	0.999937	0.997175	0.994371

Validation performance comparison:

	Bagging Tuned with Random search	Random Forest Tuned with Random search	Xgboost Tuned with Random Search
Accuracy	0.987833	0.990250	0.975000
Recall	0.856707	0.871951	0.896341
Precision	0.915309	0.945455	0.717073
F1	0.885039	0.907216	0.796748
Minimum_Vs_Model_cost	0.790361	0.812687	0.774803

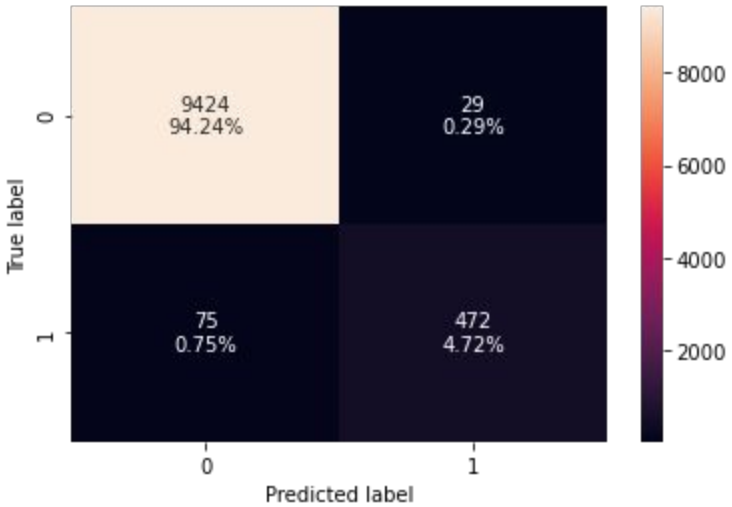
Choosing Final Model:

- Random Forest with tuned hyperparameters using the Over Sampled data has given us the best metrics between training and validation as well as one of the highest min vs model cost scores of 0.81.

- The test data (10,000) will be used with the model to build a pipeline to put into production.

- The test data was isolated from training and validation data in order to not have any data leakage from model building affect/influence the test data and final production model.

Final Model: Random Forest tuned on Over Sampled Data:

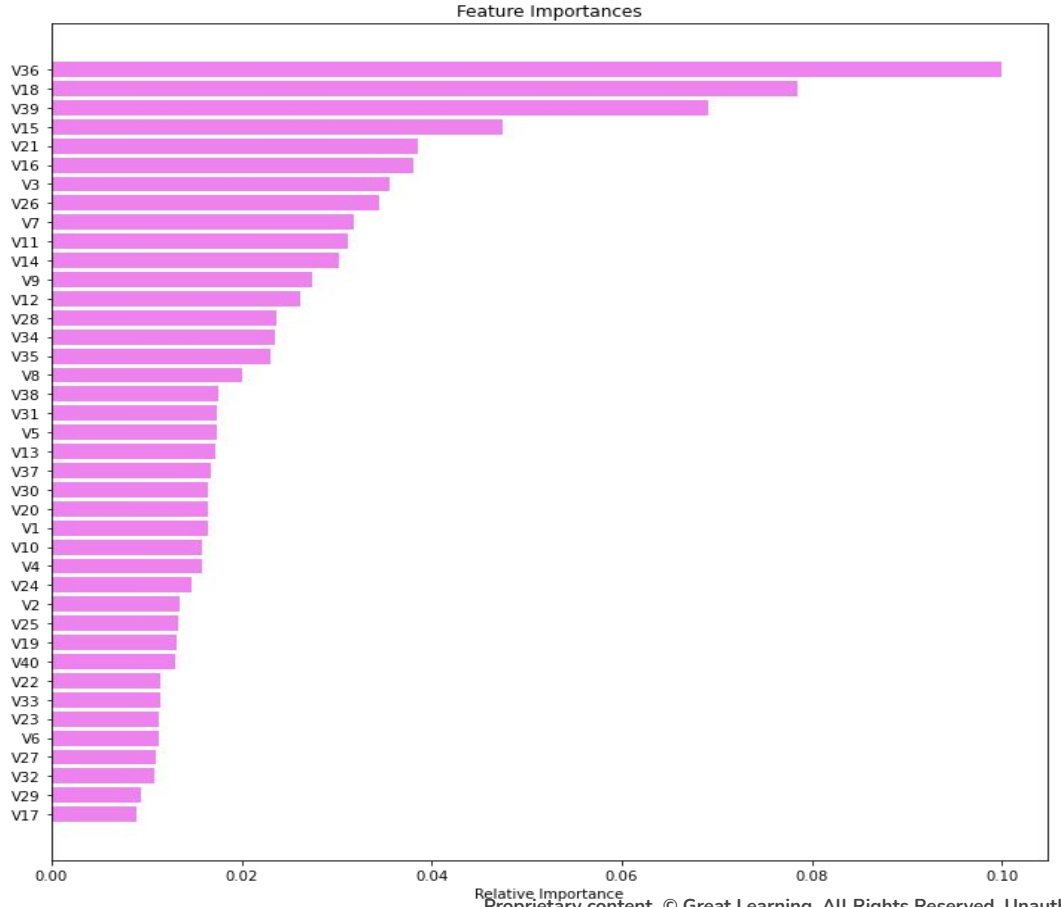


Test performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.9896	0.862888	0.942116	0.900763	0.802445

- Our final model offers very good overall metrics, a good 94.24% Identification of non failing equipment as well as a min vs model cost of an acceptable 0.80.

Final Model Feature Importance:



- Our final model places the most importance on predictors V36, V18, V39 and V15

Pipeline to Build Final Model:

Column Transformer:

- We know that we can use pipelines to standardize the model building, but the steps in a pipeline are applied to each and every variable - how can we personalize the pipeline to perform different processing on different columns
- Column transformer allows different columns or column subsets of the input to be transformed separately and the features generated by each transformer will be concatenated to form a single feature space. This is useful for heterogeneous or columnar data, to combine several feature extraction mechanisms or transformations into a single transformer
- We are doing missing value imputation for the whole data, so that if there is any missing value in the data in future that can be taken care of.
- For numerical columns, we will do missing value imputation as pre-processing

Pipeline has imputer function built in to apply column median values to any numerical values missing in future data sets.

```
Pipeline(steps=[('pre',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('num',
                                                  Pipeline(steps=[('imputer',
                                                                   SimpleImputer(strategy='median'))]),
                                                  ['V1', 'V2', 'V3', 'V4', 'V5',
                                                   'V6', 'V7', 'V8', 'V9',
                                                   'V10', 'V11', 'V12', 'V13',
                                                   'V14', 'V15', 'V16', 'V17',
                                                   'V18', 'V19', 'V20', 'V21',
                                                   'V22', 'V23', 'V24', 'V25',
                                                   'V26', 'V27', 'V28', 'V29',
                                                   'V30', ...])])),
                ('RandomForest',
                 RandomForestClassifier(max_features='log2',
                                        max_samples=0.5000000000000001,
                                        n_estimators=250, random_state=1))])
```

Business Insights for ReneWind:

- ReneWind can apply this model using the provided pipeline to future data sets.
- The model has a predictive cost of ~20% away from the minimum maintenance cost as the min vs model cost is 0.80.
- The model places the most emphasis on predictors; V36, V18, V35, V15.
- ReneWind should pay most attention to those predicators going forward and use those as an early detection system for increasing maintenance schedules in order to decrease replacement costs.
- In order to build a more robust model and potentially reach the minimum maintenance cost threshold more information will have to be gathered from ReneWind (or ciphered and translated in a way so as to not breach confidentiality).
- More information on the sensors themselves could help build a better model:
  - Where are they placed on the equipment? Outside? Inside?
  - What environment is the equipment in?
  - Does environment affect the sensor lifespan?
  - What is the repair history of the equipment?
  - Is the equipment repaired with new or used like-for-like equipment?
  - What is the normal repair schedule?