RENEWIND

Model Tuning
PGP - Data Science and Business Analytics
Arsalaan B. Saiyed

Date – 22nd July 2022

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Contents / Agenda

- Executive Summary
- Business Problem Overview and Solution Approach
- Data Background and Contents
- EDA Results
- Data Preprocessing
- Model Performance Summary
  - Model Building, Tuning & Selection
  - Final Model Testing
  - Building ML Pipeline
- Appendices
  - Appendix-A: EDA Results
  - Appendix-B: Model Performance Summary

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Executive Summary

- ***Insights & Recommendations***:

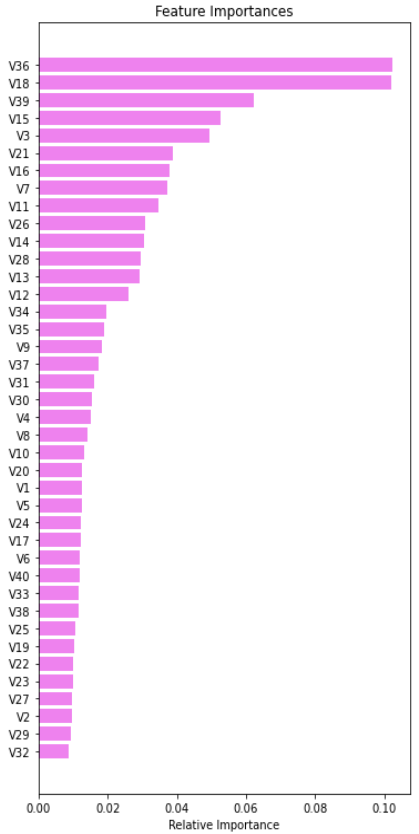
Based on the insights from EDA and building classification models to predict the likelihood of a generator for a wind turbine to fail we would like to provide some insights and recommendations.

On studying the **feature importances** and plotting some of the estimator ***decision trees*** of the final tuned random forest model with the highest recall and accuracy we would like to propose the following:

I. Variable 36 and 18 (V36,V18) were found to be the top if not one of the top nodes of most of decision tree model and were ranked number one and two in our feature importance list. Therefore, it would be crucial to monitor these features and request for *immediate preventive maintenance or repairs* if one of the readings were unusual for these variables to avoid replacement costs.

II. V39,15,3,21,16,7 &11 are important features according to the findings in the feature importance list so we would suggest for inspection of these items if the cost of replacing is high so you can prevent any replacements.

III. Another suggestion would be to maintain few more fields like dates for when each generator turbine such as ‘generator installation date’, ‘generator failure date(needing replacement)’, ‘number of breakdowns since installation’. Doing this we could predict when a generator is likely due for an inspection, repair or breakdown and extend the lifecycle of a generator as much as possible which would help forecast costs and maintain inventory to minimize as much loss as possible.



Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Business Problem Overview and Solution Approach

- ***Problem Overview:***

  "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.

  The objective is to build the best machine learning model that will help identify failures so that the generators could be repaired before failing/breaking to reduce the overall maintenance cost. The nature of predictions made by the classification model will translate as follows:

  - True positives (TP) are failures correctly predicted by the model. These will result in repairing costs.
  - False negatives (FN) are real failures where there is no detection by the model. These will result in replacement costs.
  - False positives (FP) are detections where there is no failure. These will result in inspection costs.

  It is given that the cost of repairing a generator is much less than the cost of replacing it, and the cost of inspection is less than the cost of repair.

- ***Solution Approach:***

  - Data Overview & EDA
  - Data pre-processing
  - Model Building & Comparison
  - Model Tuning & Testing
  - Model Pipeline set-up
  - Executive Summary - Actionable insights and summary

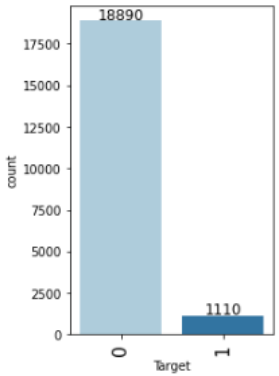
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Data Background and Contents

ReneWind provided us with two data sets. Training data and testing data both with 40 variables numbered from V1 to V40 and the target variables; “1” in the target variables should be considered as “failure” and “0” represents “No failure”.

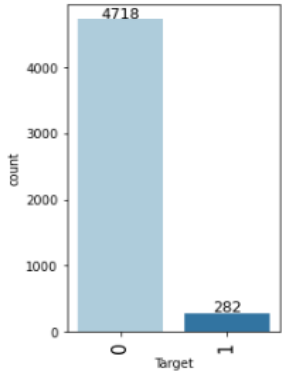
- ***Training Data***:
  - V1 & V2 had 18 missing values each

Observations	Variables
20000	41



- ***Testing Data***:
  - V1 had 5 missing values
  - V2 had 6 missing values

Observations	Variables
5000	41



Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

EDA Results

Upon conducting a exploratory data analysis of the raw data for all the 40 variables that whose readings are obtained by the sensors on the wind turbines we see that all the variables have a **normal distribution** and there was **no significant skewness or outliers** detected in any of the data except in the target variable. Since information about the actual variables are is confidential we cant derive much insight about the relations or significance of information from the visual analysis but we nevertheless display the statistical summary. *Few of the graphs plotted are attached in the appendix.*

Note: The mean, min, max etc. for all columns are given in the statistical summary in a table in Appendix-A. Also, note that bivariate analysis is also not possible as we don't know which of the variables to pair together for analysis or what is their significance. Building a *heat map* we did see quite a few variables had a strong correlation with each other. The visual is attached in the Appendix slide for reference.

Link to Appendix slide on data background check

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Data Preprocessing

- Duplicate value check
- Missing value treatment
- Outlier check (treatment if needed)
- Feature engineering
- Data preparation for modeling

***Note**: You can use more than one slide if needed*

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Data Preprocessing

Let's take a quick look at a small sample of our data. This will help us get some idea of the attributes of the data and also help us understand the degree of cleaning needed before we can build a model.

Random sample of the data (5 rows, 41 columns):

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21
0	-4.465	-4.679	3.102	0.506	-0.221	-2.033	-2.911	0.051	-1.522	3.762	-5.715	0.736	0.981	1.418	-3.376	-3.047	0.306	2.914	2.270	4.395	-2.388
1	3.366	3.653	0.910	-1.368	0.332	2.359	0.733	-4.332	0.566	-0.101	1.914	-0.951	-1.255	-2.707	0.193	-4.769	-2.205	0.908	0.757	-5.834	-3.065
2	-3.832	-5.824	0.634	-2.419	-1.774	1.017	-2.099	-3.173	-2.082	5.393	-0.771	1.107	1.144	0.943	-3.164	-4.248	-4.039	3.689	3.311	1.059	-2.143
3	1.618	1.888	7.046	-1.147	0.083	-1.530	0.207	-2.494	0.345	2.119	-3.053	0.460	2.705	-0.636	-0.454	-3.174	-3.404	-1.282	1.582	-1.952	-3.517
4	-0.111	3.872	-3.758	-2.983	3.793	0.545	0.205	4.849	-1.855	-6.220	1.998	4.724	0.709	-1.989	-2.633	4.184	2.245	3.734	-6.313	-5.380	-0.887

V22	V23	V24	V25	V26	V27	V28	V29	V30	V31	V32	V33	V34	V35	V36	V37	V38	V39	V40	Target
0.646	-1.191	3.133	0.665	-2.511	-0.037	0.726	-3.982	-1.073	1.667	3.060	-1.690	2.846	2.235	6.667	0.444	-2.369	2.951	-3.480	0
1.597	-1.757	1.766	-0.267	3.625	1.500	-0.586	0.783	-0.201	0.025	-1.795	3.033	-2.468	1.895	-2.298	-1.731	5.909	-0.386	0.616	0
1.650	-1.661	1.680	-0.451	-4.551	3.739	1.134	-2.034	0.841	-1.600	-0.257	0.804	4.086	2.292	5.361	0.352	2.940	3.839	-4.309	0
-1.206	-5.628	-1.818	2.124	5.295	4.748	-2.309	-3.963	-6.029	4.949	-3.584	-2.577	1.364	0.623	5.550	-1.527	0.139	3.101	-1.277	0
2.062	9.446	4.490	-3.945	4.582	-8.780	-3.383	5.107	6.788	2.044	8.266	6.629	-10.069	1.223	-3.230	1.687	-2.164	-3.645	6.510	0

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Data Preprocessing

- Duplicate Value Check: No duplicate entries were found in the given data
- Missing Value Treatment:

  Training Data had 18 missing values in V1 & V2 – We treat the missing values by imputing them with the medians of the variables V1&2

  Testing Data had 5 missing values in V1 and 6 in V2 – We treat the missing values by imputing them with the medians of the variables V1&2
- Outlier Check:

  Since we don't have the information and knowledge about what the variables actually are we cant treat any outliers and also note that EDA shows that there are no extreme outliers in any of the variables. Also, since this data was extracted from the sensors installed on the turbines we can assume that all the variables have accurate real readings.
- Feature Engineering:

  There was no need for feature engineering.

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Model Performance Summary

- Overview of ML model and its parameters
- Model building
- Model tuning
- Model comparison
- Final model selection & testing
- Building pipeline for final model

Link to Appendix slide on model assumptions

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Model Performance Summary

- Overview of ML model:

  ReneWind requires an intuitive model to predict based on the different variable readings provided by the sensors on their wind turbine which of the turbines are likely to fail. This is a classification problem so we will train and test ML models to find the best model for ReneWind.

- ML model parameters:

  - **Model Building**: We split the 'Training' Data given to us into two parts – **Training and Validation Data** in a *75:25* ratio then, train six models for classification on 3 different types of training and validation data listed below and compare the results of **the 18 Models**. Models Used: *Logistic Regression, Bagging, Random Forest, Gradient Boosting, Ada Boosting, XG Boost*

    a. Original Data

    b. Over-sampled Data

    c. Under-sampled Data

  - **Model Evaluation Criterion**: The nature of predictions made by the classification model will translate as follows:

    a. True positives (TP) are failures correctly predicted by the model.

    b. False negatives (FN) are real failures in a generator where there is no detection by model.

    c. False positives (FP) are failure detections in a generator where there is no failure.

    Which case is more important?

    - We need to choose the metric which will ensure that the maximum number of generator failures are predicted correctly by the model.
    - We would 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.

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Model Performance Summary

- Model Building:
  a. Original Data: Using the data of 20,000 records as given to us by ReneWind in the 'Train.csv' file we trained 6 of the selected models and obtained *recall scores* for all. **XG Boost** was found to be the best performing model with the highest recall score on the validation data set.
  b. Over-sampled Data: Before oversampling we had only 5.5% data for the failure scenario of the turbines which meant less data. Using SMOTE (Synthetic Minority Over Sampling Technique) we over sampled our data giving us a total of 28,336 records of which 14,168 were now for failure label and 14,168 for healthy label, before oversampling the count was 832 and 14,168 for the failure and healthy labels respectively.

     We trained the exact same models on our over-sampled data set and found **Gradient Boost** Model to have the highest *recall scores on the validation data set.*
  a. Under-sampled Data: As opposed to overpopulating our data set in over-sampling we scale down the records for the healthy label records to that of the failure label records and we are left with 832 to samples for each label. Training our models yet again on our under-sampled data we see that **XG Boost** has the best *recall score* on the validation data set.

- Model Tuning:

  We tuned the hyperparameters of the following models based on the performance factors like recall score, accuracy and consistency between the train and validation set on the 3 different data sets(Their performance is provided in the *Appendix*):

  I. *AdaBoost* w/ Over-sampled Data
  II. *Random Forest* w/ Under-sampled Data
  III. *Gradient Boost* w/ Over-sampled Data
  IV. *XG Boost* w/ Over-sampled Data

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Model Performance Summary

- Model Comparison:

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	1.000
Recall	0.993	0.988	0.933	1.000
Precision	0.994	0.995	0.989	1.000
F1	0.993	0.992	0.960	1.000

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	0.971	0.979	0.938	0.986
Recall	0.845	0.849	0.885	0.878
Precision	0.693	0.789	0.468	0.878
F1	0.762	0.818	0.612	0.878

Conclusion:

Since our objective is to maximize recall we can note that the highest recall on the validation data set was for the tuned random forest with under-sampled data(Recall = 0.885)

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Model Performance Summary

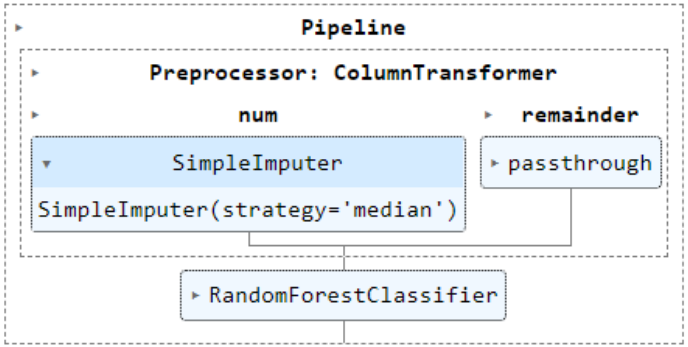
- Final Model Testing:

  The model that was selected was the **Tuned Random Forest** as it had the highest recall score compared to all the other tuned model. The model was then tested against the unseen test data provided by ReneWind in the 'Test.csv' file and it yielded a recall score of 0.879 with an accuracy of 0.944. Other performance metrics are provided in the *appendix*.

- Building Model Pipeline:

  Then using Pipeline from sklearn library we built a the pipeline for our best model the tuned random forest model with the imputer step as the preprocessor step. Since we had only numerical values we did not need to impute any categorical features.

  Trained and tested the *pipeline* and we got a recall score of 0.872.



Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

APPENDIX A – EDA Results

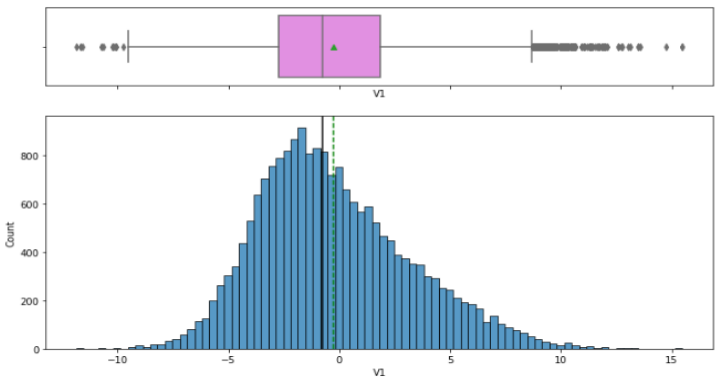
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

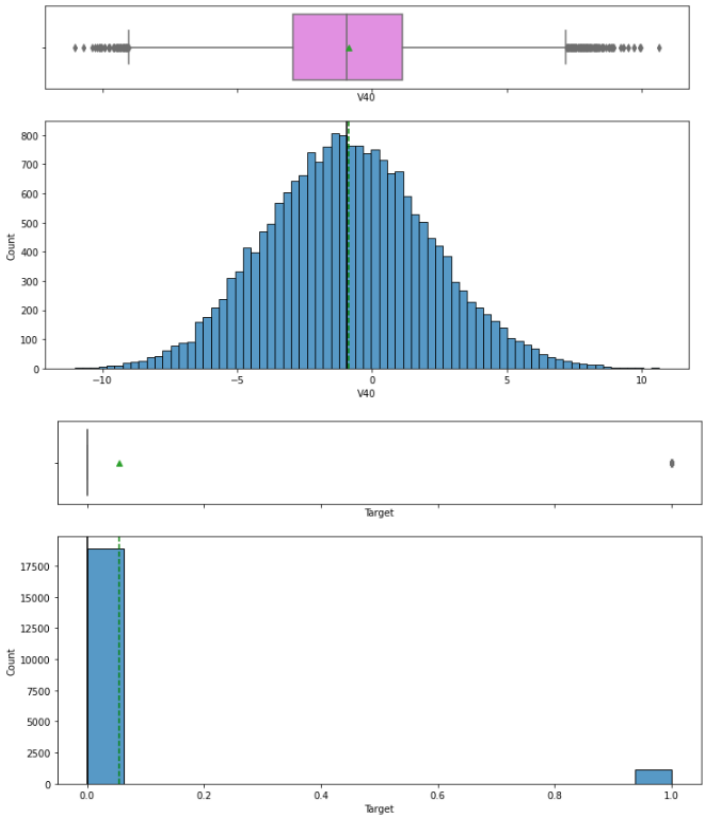
APPENDIX A: Statistical Summary

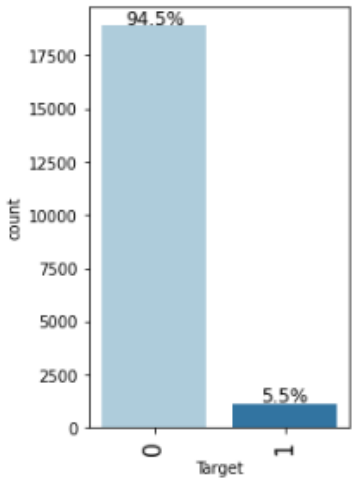
	count	mean	std	min	25%	50%	75%	max
V1	19982.000	-0.272	3.442	-11.876	-2.737	-0.748	1.840	15.493
V2	19982.000	0.440	3.151	-12.320	-1.641	0.472	2.544	13.089
V3	20000.000	2.485	3.389	-10.708	0.207	2.256	4.566	17.091
V4	20000.000	-0.083	3.432	-15.082	-2.348	-0.135	2.131	13.236
V5	20000.000	-0.054	2.105	-8.603	-1.536	-0.102	1.340	8.134
V6	20000.000	-0.995	2.041	-10.227	-2.347	-1.001	0.380	6.976
V7	20000.000	-0.879	1.762	-7.950	-2.031	-0.917	0.224	8.006
V8	20000.000	-0.548	3.296	-15.658	-2.643	-0.389	1.723	11.679
V9	20000.000	-0.017	2.161	-8.596	-1.495	-0.068	1.409	8.138
V10	20000.000	-0.013	2.193	-9.854	-1.411	0.101	1.477	8.108
V11	20000.000	-1.895	3.124	-14.832	-3.922	-1.921	0.119	11.826
V12	20000.000	1.605	2.930	-12.948	-0.397	1.508	3.571	15.081
V13	20000.000	1.580	2.875	-13.228	-0.224	1.637	3.460	15.420
V14	20000.000	-0.951	1.790	-7.739	-2.171	-0.957	0.271	5.671
V15	20000.000	-2.415	3.355	-16.417	-4.415	-2.383	-0.359	12.246
V16	20000.000	-2.925	4.222	-20.374	-5.634	-2.683	-0.095	13.583
V17	20000.000	-0.134	3.345	-14.091	-2.216	-0.015	2.069	16.756
V18	20000.000	1.189	2.592	-11.644	-0.404	0.883	2.572	13.180
V19	20000.000	1.182	3.397	-13.492	-1.050	1.279	3.493	13.238
V20	20000.000	0.024	3.669	-13.923	-2.433	0.033	2.512	16.052
V21	20000.000	-3.611	3.568	-17.956	-5.930	-3.533	-1.266	13.840
V22	20000.000	0.952	1.652	-10.122	-0.118	0.975	2.026	7.410
V23	20000.000	-0.366	4.032	-14.866	-3.099	-0.262	2.452	14.459
V24	20000.000	1.134	3.912	-16.387	-1.468	0.969	3.546	17.163
V25	20000.000	-0.002	2.017	-8.228	-1.365	0.025	1.397	8.223
V26	20000.000	1.874	3.435	-11.834	-0.338	1.951	4.130	16.836
V27	20000.000	-0.612	4.369	-14.905	-3.652	-0.885	2.189	17.560
V28	20000.000	-0.883	1.918	-9.269	-2.171	-0.891	0.376	6.528
V29	20000.000	-0.986	2.684	-12.579	-2.787	-1.176	0.630	10.722
V30	20000.000	-0.016	3.005	-14.796	-1.867	0.184	2.036	12.506
V31	20000.000	0.487	3.461	-13.723	-1.818	0.490	2.731	17.255
V32	20000.000	0.304	5.500	-19.877	-3.420	0.052	3.762	23.633
V33	20000.000	0.050	3.575	-16.898	-2.243	-0.066	2.255	16.692
V34	20000.000	-0.463	3.184	-17.985	-2.137	-0.255	1.437	14.358
V35	20000.000	2.230	2.937	-15.350	0.336	2.099	4.064	15.291
V36	20000.000	1.515	3.801	-14.833	-0.944	1.567	3.984	19.330
V37	20000.000	0.011	1.788	-5.478	-1.256	-0.128	1.176	7.467
V38	20000.000	-0.344	3.948	-17.375	-2.988	-0.317	2.279	15.290
V39	20000.000	0.891	1.753	-6.439	-0.272	0.919	2.058	7.760
V40	20000.000	-0.876	3.012	-11.024	-2.940	-0.921	1.120	10.654
Target	20000.000	0.056	0.229	0.000	0.000	0.000	0.000	1.000

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

APPENDIX A: Univariate Analysis





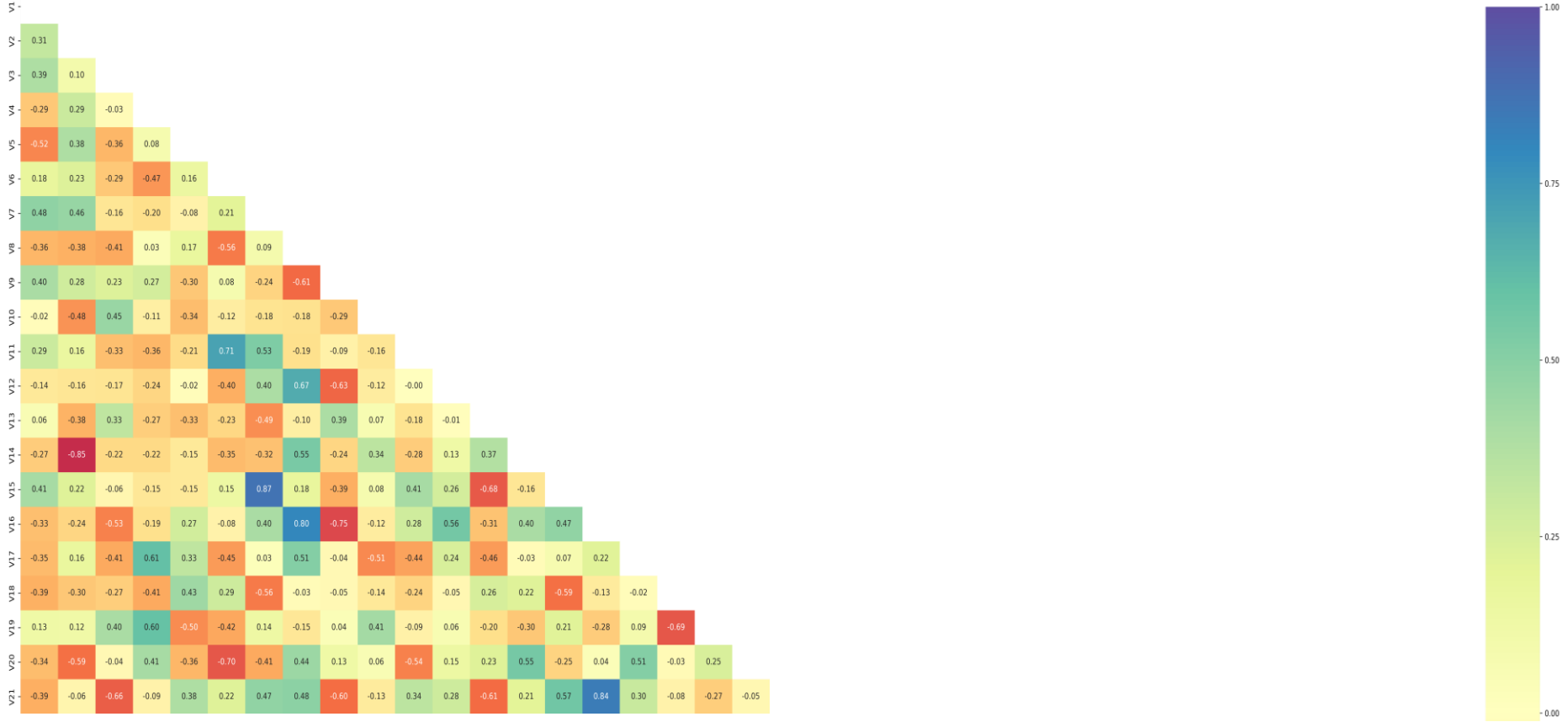


Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

APPENDIX A – EDA Results (Bivariate Analysis)

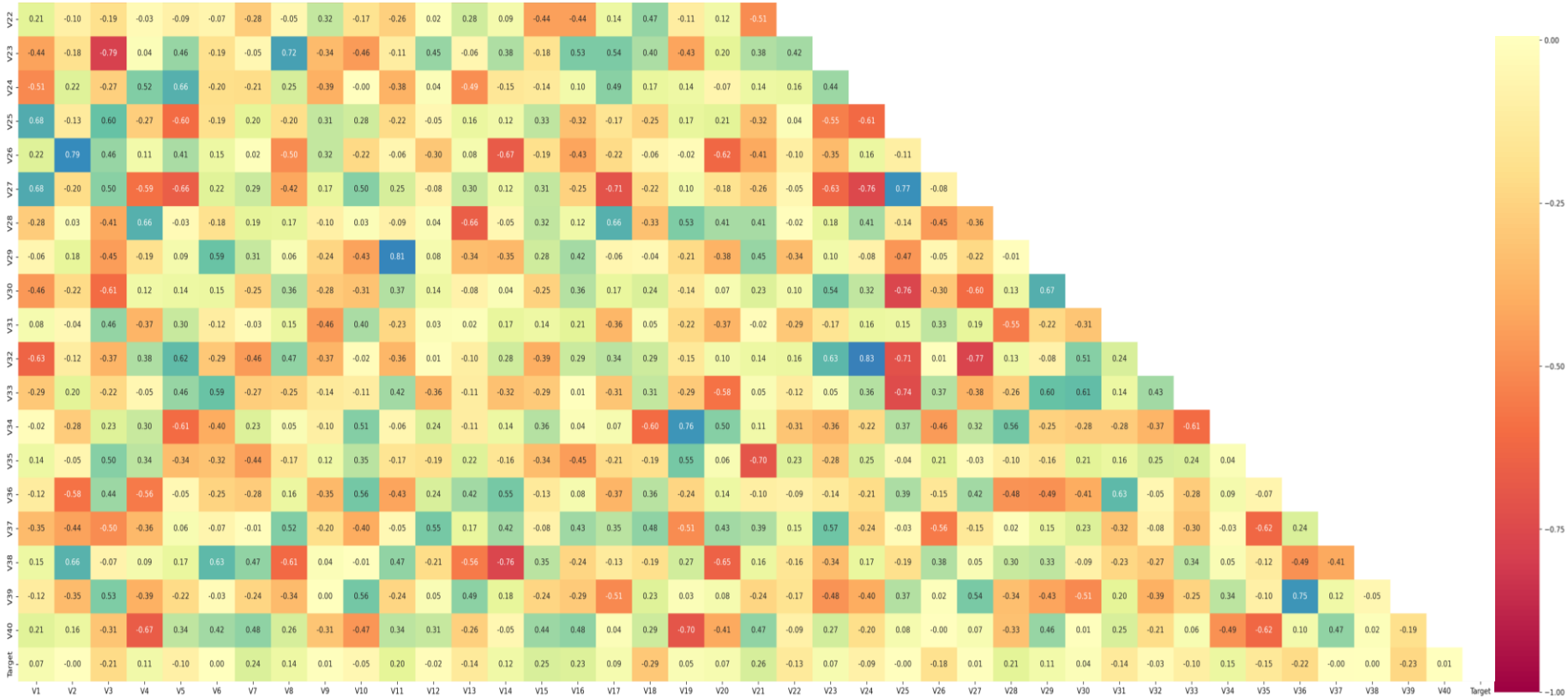
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

APPENDIX A: Bivariate Analysis



Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

APPENDIX A: Bivariate Analysis



Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

APPENDIX B – Model Performance Summary

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

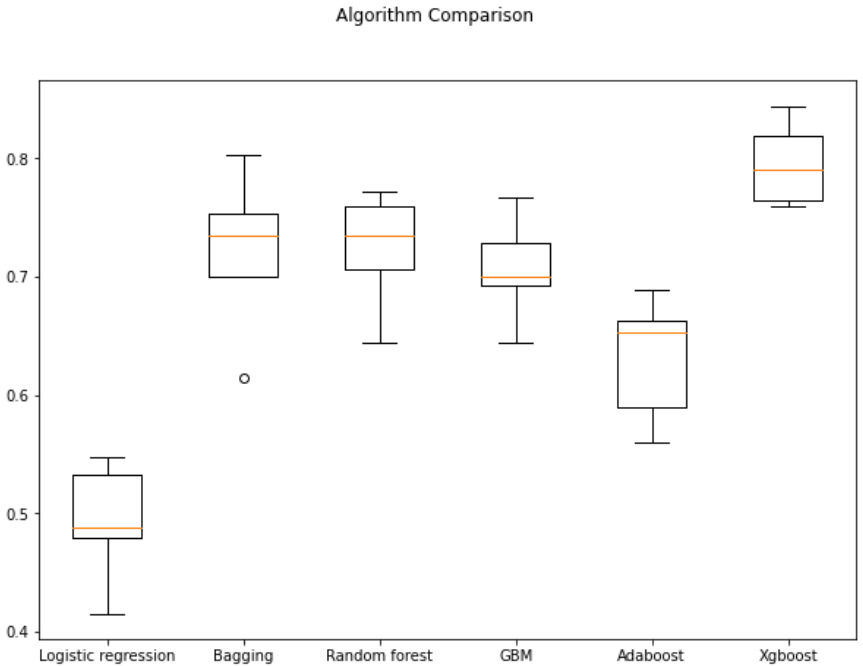
Original Data Model Comparison

```
Cross-Validation Cost:

Logistic regression: 0.4927566553639709
Bagging: 0.7210807301060529
Random forest: 0.7235192266070268
GBM: 0.7066661857008874
Adaboost: 0.6309140754635308
Xgboost: 0.7956208065796118

Validation Performance:

Logistic regression: 0.48201438848920863
Bagging: 0.7302158273381295
Random forest: 0.7266187050359713
GBM: 0.7230215827338129
Adaboost: 0.6762589928057554
Xgboost: 0.8201438848920863
Wall time: 3min 51s
```



Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

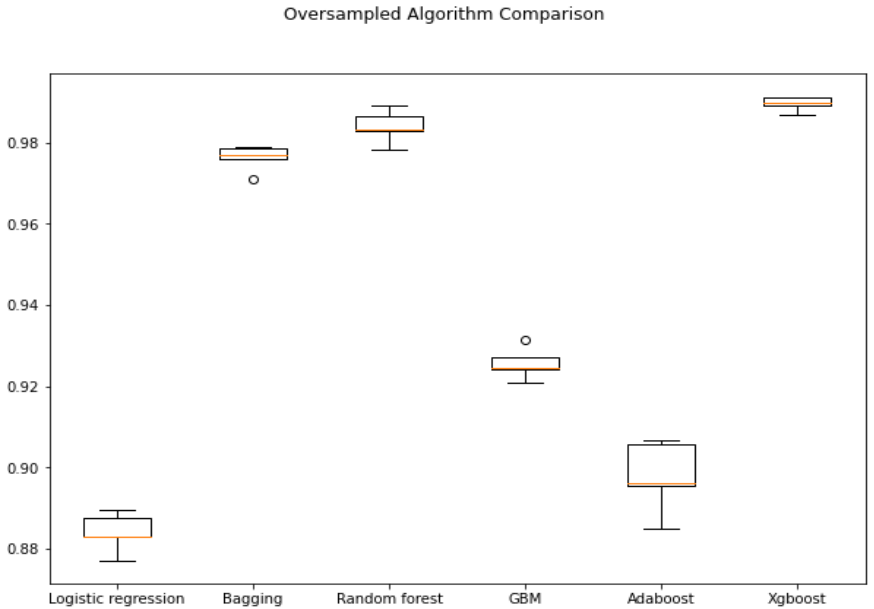
Over-Sampled Data Model Comparison

```
Cross-Validation Cost:

Logistic regression: 0.883963699328486
Bagging: 0.9762141471581656
Random forest: 0.9839075260047615
GBM: 0.9256068151319724
Adaboost: 0.8978689011775473
Xgboost: 0.989554053559209

Validation Performance:

Logistic regression: 0.8489208633093526
Bagging: 0.8345323741007195
Random forest: 0.8489208633093526
GBM: 0.8776978417266187
Adaboost: 0.8561151079136691
Xgboost: 0.8669064748201439
Wall time: 6min 45s
```



Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

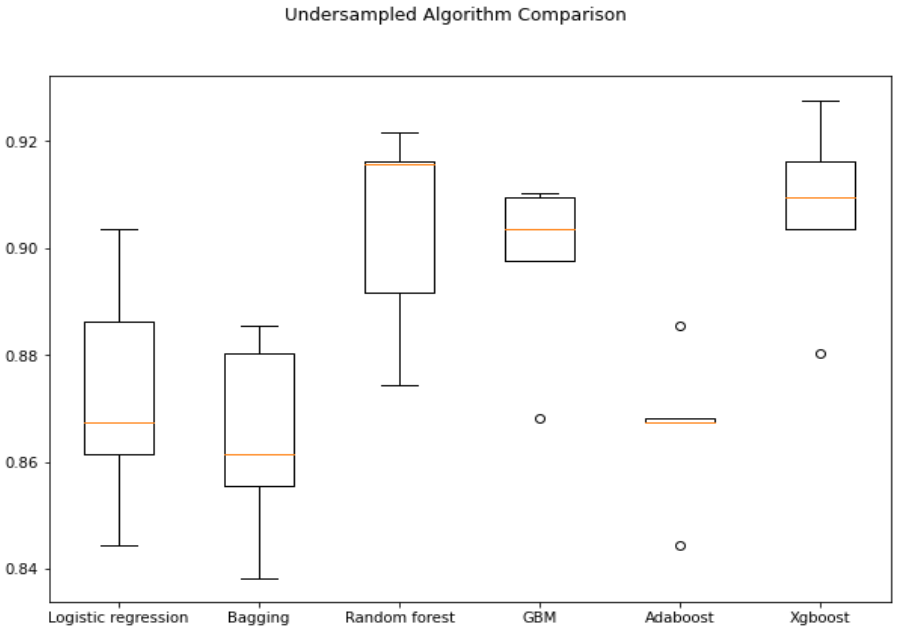
Under-Sampled Data Model Comparison

```
Cross-Validation Cost:

Logistic regression: 0.8726138085275232
Bagging: 0.8641945025611427
Random forest: 0.9038669648654498
GBM: 0.8978572974532861
Adaboost: 0.8666113556020489
Xgboost: 0.9074742082100858

Validation Performance:

Logistic regression: 0.8525179856115108
Bagging: 0.8705035971223022
Random forest: 0.8920863309352518
GBM: 0.8884892086330936
Adaboost: 0.8489208633093526
Xgboost: 0.9028776978417267
Wall time: 18 s
```



Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Tuned Ada Boost using over-sampled data

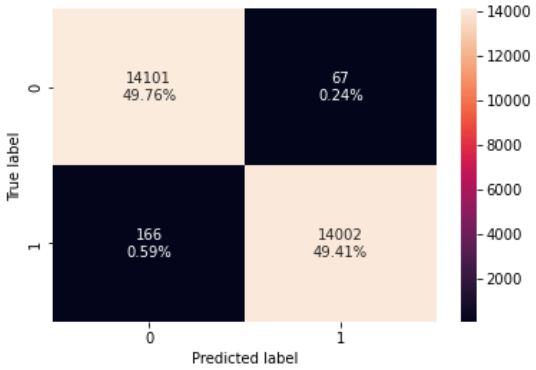
```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=3,
                                                         random_state=1),
                   learning_rate=0.2, n_estimators=200)
```

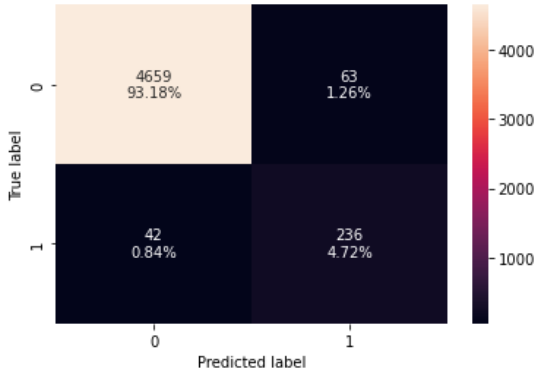
base_estimator: DecisionTreeClassifier

```
DecisionTreeClassifier(max_depth=3, random_state=1)
```

DecisionTreeClassifier

```
DecisionTreeClassifier(max_depth=3, random_state=1)
```





AdaBoost(Oversampled Data)
Training Perfromance:

	Accuracy	Recall	Precision	F1
0	0.992	0.988	0.995	0.992

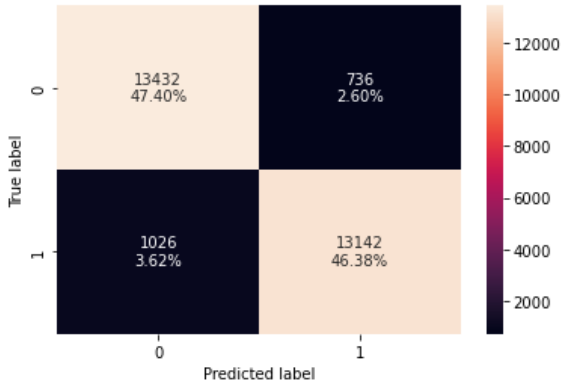
AdaBoost(Oversampled Data)
Testing Perfromance:

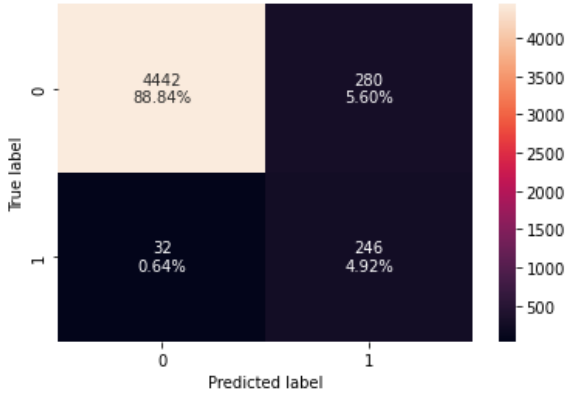
	Accuracy	Recall	Precision	F1
0	0.979	0.849	0.789	0.818

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Tuned Random Forest using under-sampled data

```
RandomForestClassifier
RandomForestClassifier(max_samples=0.5, min_samples_leaf=2, n_estimators=300,
                       random_state=1)
```





Random Forest(Undersampled Data)
Training Perfromance:

	Accuracy	Recall	Precision	F1
0	0.961	0.933	0.989	0.960

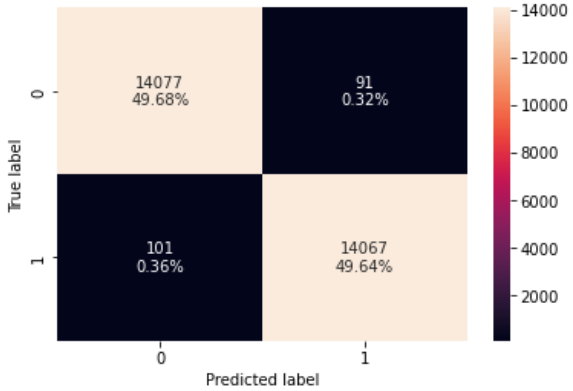
Random Forest(Undersampled Data)
Testing Perfromance:

	Accuracy	Recall	Precision	F1
0	0.938	0.885	0.468	0.612

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

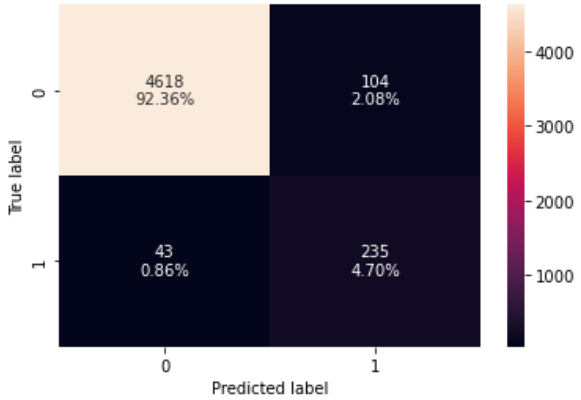
Tuned Gradient Boost using over-sampled data

```
GradientBoostingClassifier
GradientBoostingClassifier(learning_rate=1, max_features=0.5, n_estimators=125,
                           random_state=1, subsample=0.7)
```



Gradient Boost (Oversampled Data)
Training Perfromance:

	Accuracy	Recall	Precision	F1
0	0.993	0.993	0.994	0.993



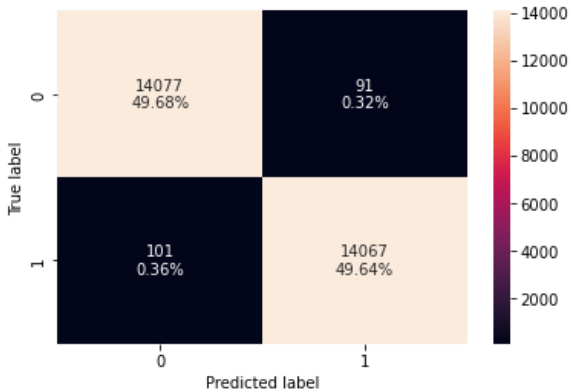
Gradient Boost (Oversampled Data)
Testing Perfromance:

	Accuracy	Recall	Precision	F1
0	0.971	0.845	0.693	0.762

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

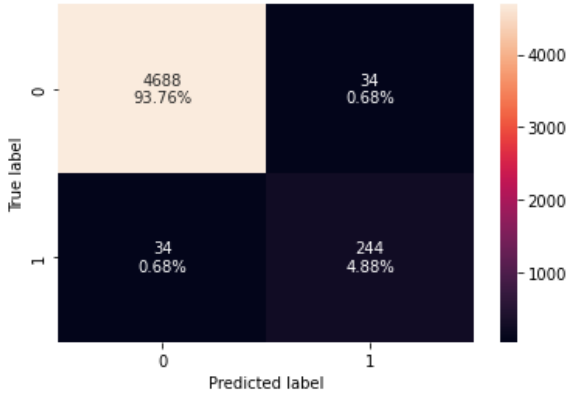
XG Boost using over-sampled data

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              eval_metric='logloss', gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.2, 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, predictor='auto', random_state=1,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=10, subsample=0.8,
              tree_method='exact', validate_parameters=1, verbosity=None)
```



XG Boost (Oversampled Data)
Training Perfromance:

	Accuracy	Recall	Precision	F1
0	1.000	1.000	1.000	1.000



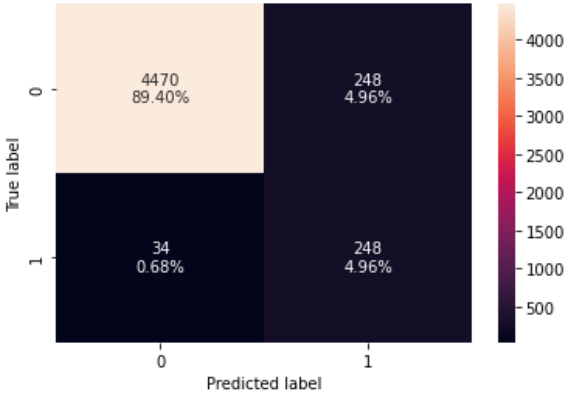
XG Boost (Oversampled Data)
Testing Perfromance:

	Accuracy	Recall	Precision	F1
0	0.986	0.878	0.878	0.878

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

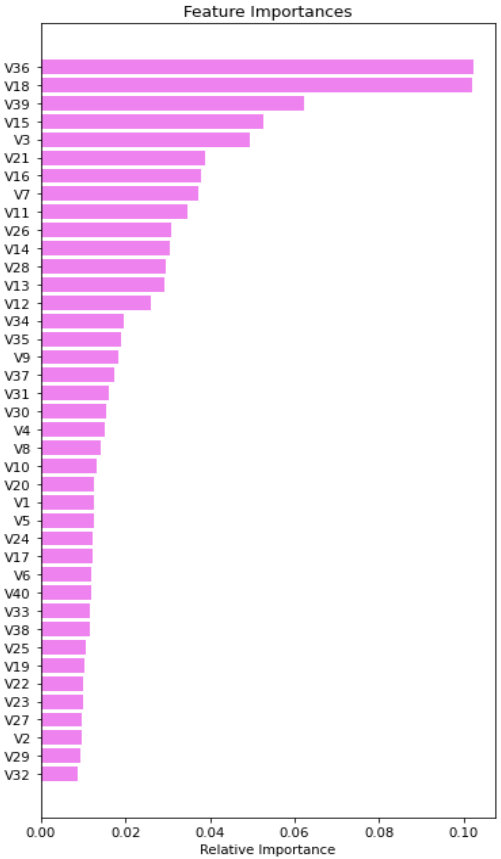
Final Model Testing: Tuned Random Forest

```
RandomForestClassifier
RandomForestClassifier(max_samples=0.5, min_samples_leaf=2, n_estimators=300,
                       random_state=1)
```



Test performance:

	Accuracy	Recall	Precision	F1
0	0.944	0.879	0.500	0.638



Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Final Model Pipeline

Pipeline

Preprocessor: ColumnTransformer

num

['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', 'V31', 'V32', 'V33', 'V34', 'V35', 'V36', 'V37', 'V38', 'V39', 'V40']

SimpleImputer

SimpleImputer(strategy='median')

remainder

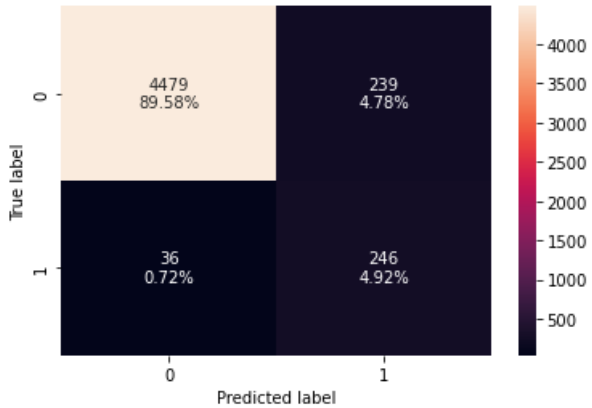
[]

passthrough

passthrough

RandomForestClassifier

RandomForestClassifier(max_samples=0.5, min_samples_leaf=2, n_estimators=300, random_state=1)

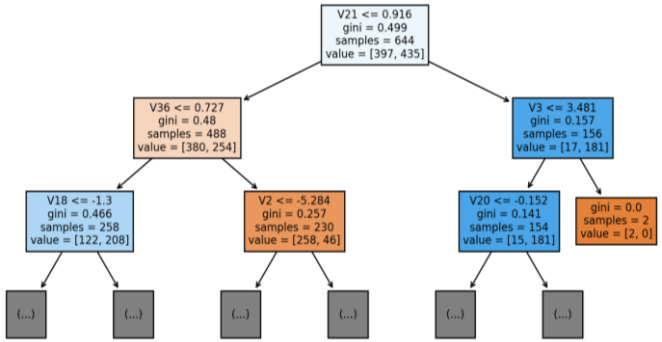


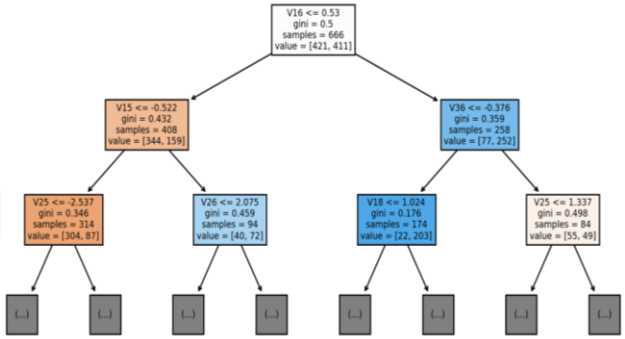
Testing Performance:

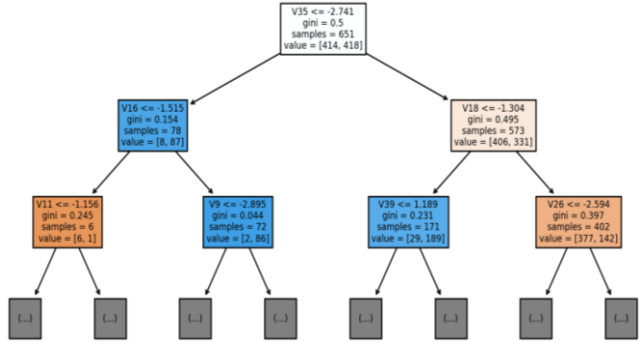
	Accuracy	Recall	Precision	F1
0	0.945	0.872	0.507	0.641

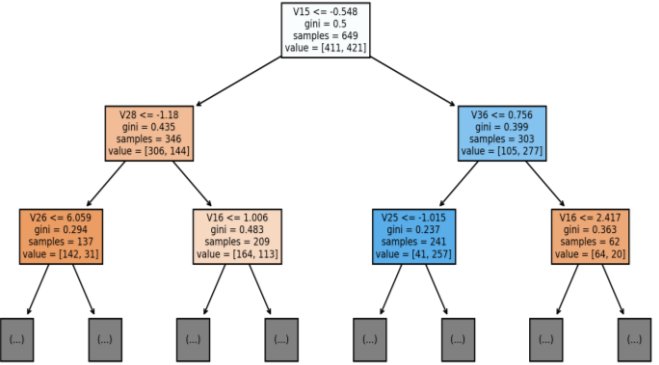
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

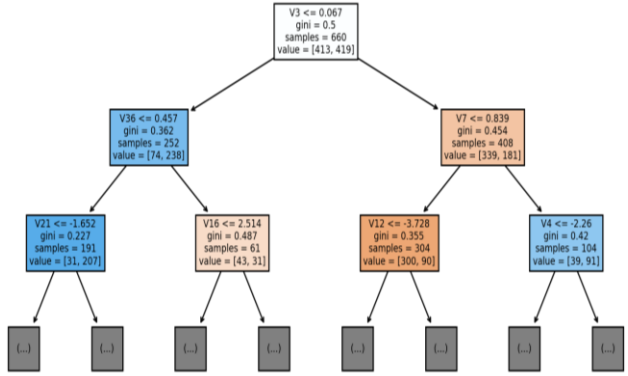
Final Mode: Decision Trees from Random Forest Model

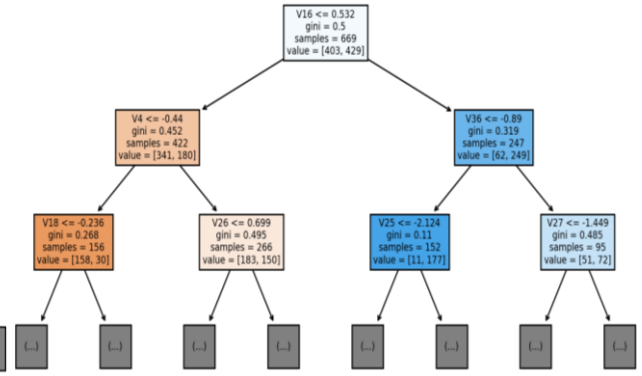












Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Great Learning

Happy Learning !