RENEWIND
PREDICTIVE MAINTENACE PROJECT

By: Syeda ambreen karim bokhari

CONTENTS

CORE BUSINESS IDEA:

- Renewable energy sources play an increasingly important role in the global environmental energy production. Wind energy is one of the most developed renewable energy technologies worldwide.
- There are sensors fitted across different machines, which are involved in the process of energy generation collect data related to various environmental factors and additional features related to various parts of the wind turbine.

BUSINESS PROBLEM OVERVIEW AND SOLUTION APPROACH

- **Problem to tackle**
  - 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. Predictive maintenance is used as 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 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.
  - A Machine Learning based solution is needed that can help in identifying failure so that overall maintenance cost can be reduced

BUSINESS PROBLEM OVERVIEW AND SOLUTION APPROACH

- **Financial implications**
  - “1” in the target variables should be considered as “failure” and “0” will represent “No failure”.
  - 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)
  - Replacement cost = $40,000, Repair cost = $15,000, Inspection cost = $5,000

BUSINESS PROBLEM OVERVIEW AND SOLUTION APPROACH

- **Financial implications**

  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.

BUSINESS PROBLEM OVERVIEW AND SOLUTION APPROACH

- **How to use ML model to solve the problem**
  - We need to analyse the provided data and build a Classification model to predict failure and to maximize the ratio of minimum possible maintenance cost and the maintenance cost associated with the model.
  - The XGBoost model is trained and tested on the available data and can be used to predict the failure and reduce the maintenance cost.

DATA OVERVIEW

Data Overview: There are two dataset files, which contain the sensor data with following specifications:

Train dataset:			
Variable name	Data types	Description	Missing Values
1. V1	numeric	float	46
2. V2	numeric	float	39
3. V3-V40	numeric	float	0
Total records: 40000		Input variables: 40	Target variable: failure:1, no-failure:0

Test dataset:			
Variable name	Data types	Description	Missing Values
1. V1	numeric	float	11
2. V2	numeric	float	07
3. V3-V40	numeric	float	0
Total records: 10000		Input variables: 40	Target variable: failure:1, no-failure:0

DATA OVERVIEW: FEATURE ENGINEERING

- Brief description of significant manipulations made to raw data

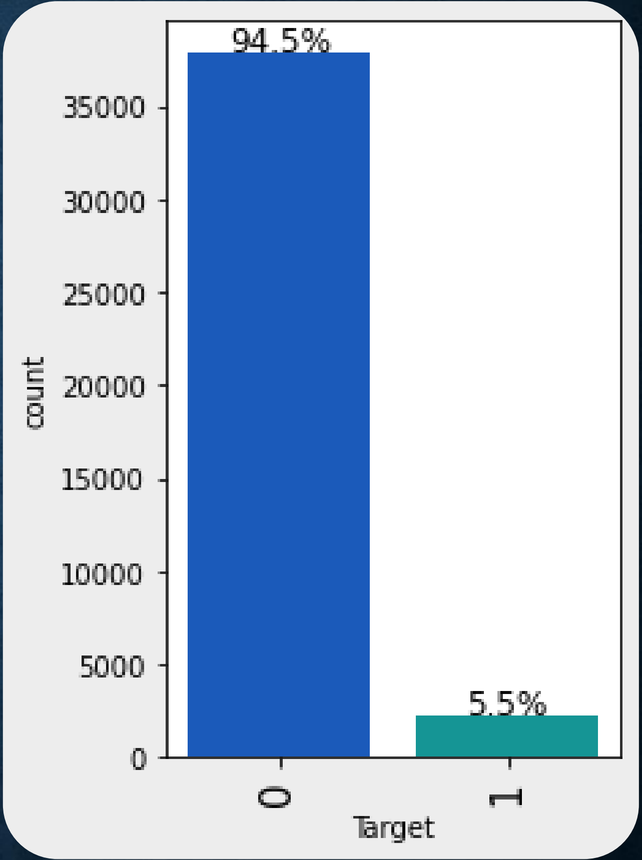
Variable name	Data description	Treatment
V1, V2	There were missing values in both of these variables in both train and test datasets.	Median imputer was used to fill missing values with median value.
There are no duplicates in the train and test data sets		

Input variables are 40 from different sensors, values of many input variables are ranging from negative to positive continuous values.

EXPLORATORY DATA ANALYSIS

UNIVARIATE ANALYSIS

- Graphs and observation about the target attribute:



<u>Observations</u>

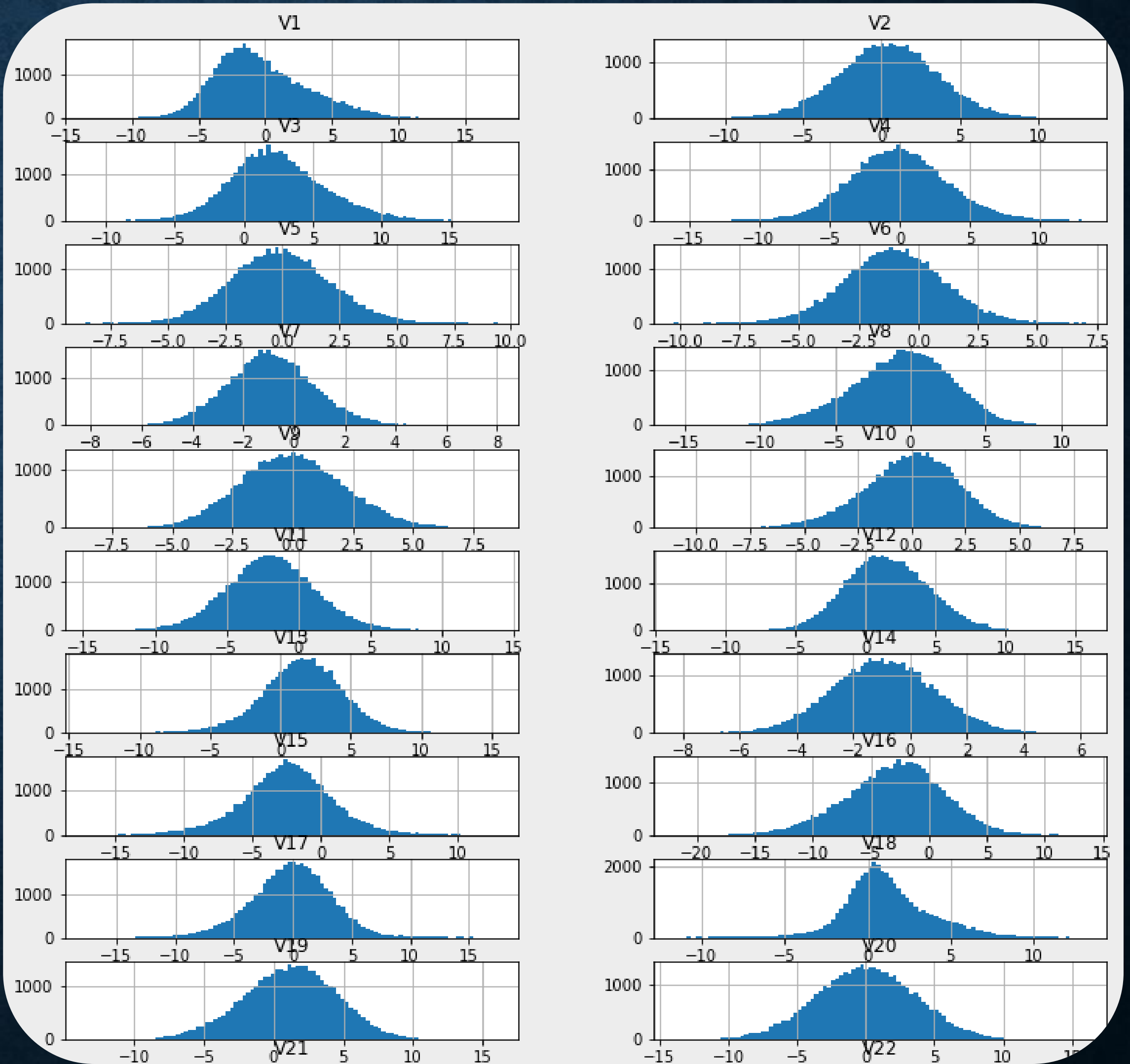
- “Target” is the target variable.
- It is a numeric Boolean variable with two values: failure: 1 and no-failure: 0
- 94.5% of the data has no failure.
- Only 5.5% of the data has failure.
- The two classes are heavily imbalanced and oversampling treatment was done to balance the dataset.

EDA

- Graphs showing the distributions of numerical variables

Observations:

- Almost all the variables have normal distributions.
- Distributions of V1, V18, V27, V37 are a bit right skewed

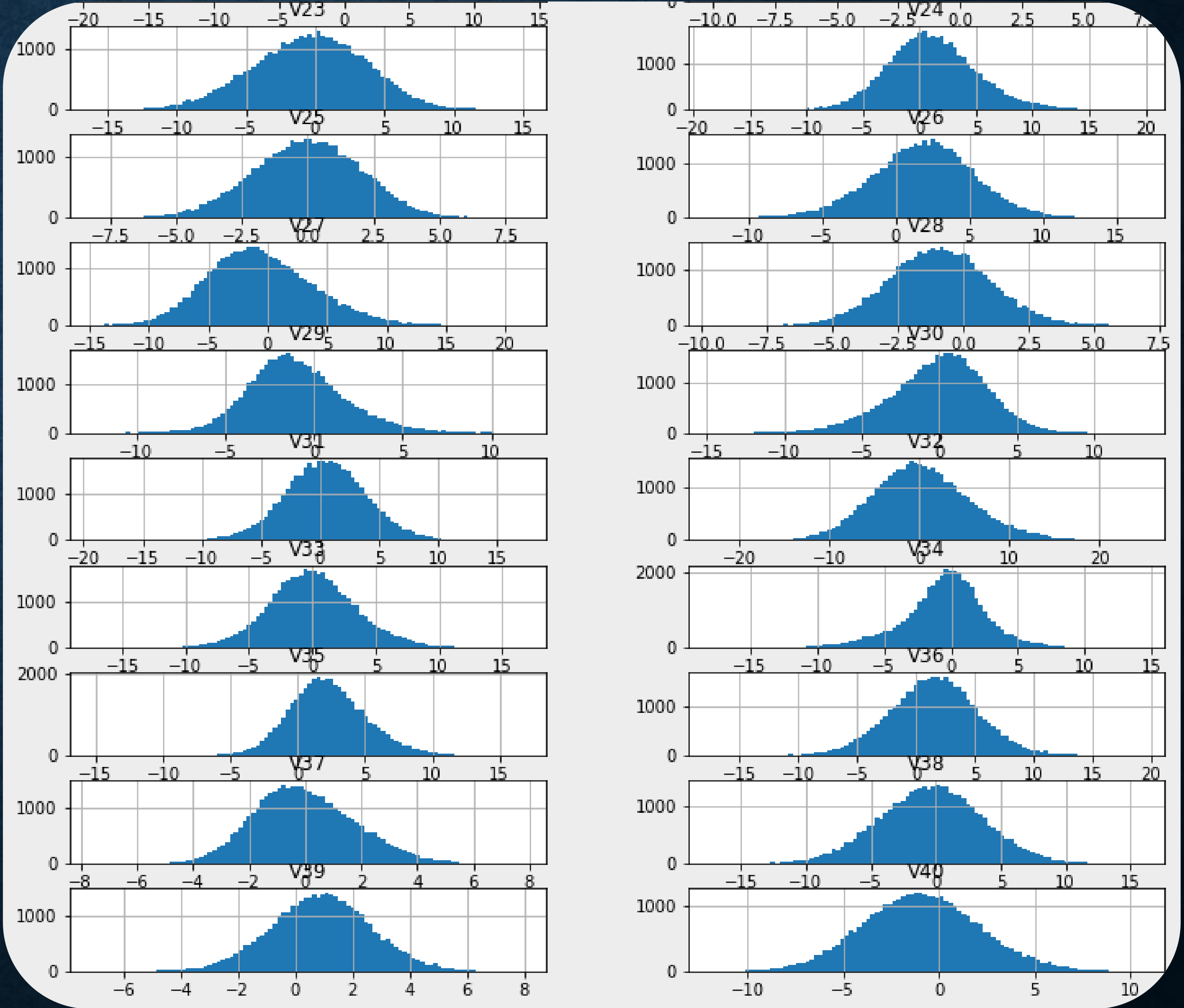


EDA

- Graphs showing the distributions of numerical variables.

Observations:

- Almost all the variables have normal distributions.
- Distributions of V1, V18, V27, V37 are a bit right skewed

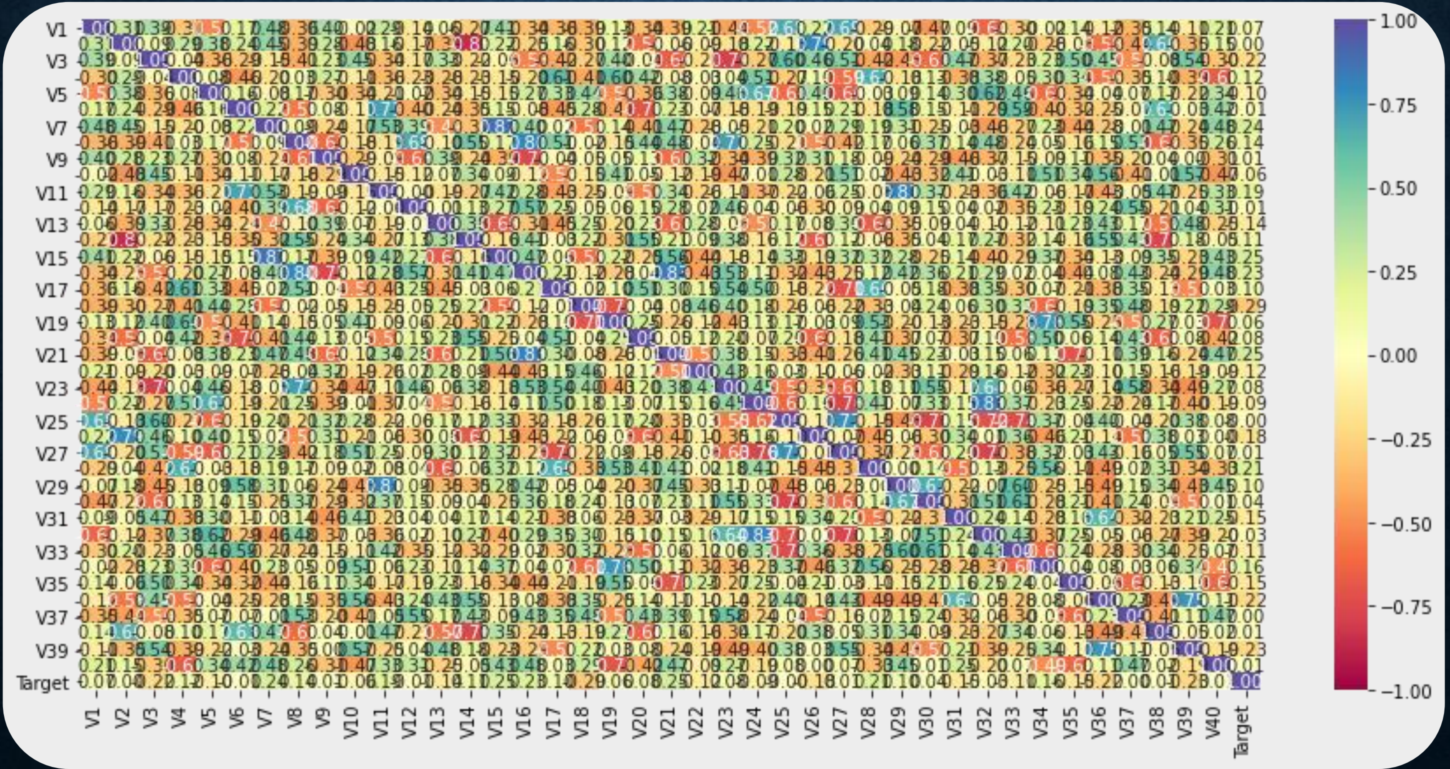


EDA

Correlation map showing association between predictors.

<u>observations:</u>

- There are some variables showing correlation like V7 and V15, V9 and V16, V23 and V32. There are others as well.
- There are too many variables to read the correlation heatmap correctly.



MODEL PERFORMANCE SUMMARY

Overview of ML model and its parameters:

- Multiple Classification models were built to
  - find dependency of target variable: Target on predictors and
  - maximize Minimum cost/Cost associated with model
  - **Total rows and columns in training data** after data preprocessing: 40000 rows × 41 columns
  - **Total rows and columns in validation data** after data preprocessing: 10000 rows × 41 columns
  - **Total rows and columns in test data** after data preprocessing: 10000 rows × 41 columns
- **Target variable:** Target
- **Predictors:** ' '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'

MODEL PERFORMANCE SUMMARY

Overview of ML model and its parameters:

- Data was divided into Train and Test at 30:70 ratio.
- **Number of rows in train data** = 30000,
- **Number of rows in test data** = 10000
- **Number of rows in test data** = 10000
- Train dataset Failure True Values : 94.5%
- Train dataset Failure False Values : 5.5%

MODEL EVALUATION CRITERION

Model can make wrong predictions as:

- Predicting a generator has failure when it is ok.
- Predicting a generator has no failure when it has failure.

Which case is more important?

- 3 types of cost are associated with the provided problem
- Replacement cost - False Negatives - Predicting no failure, while there will be a failure
- Inspection cost - False Positives - Predicting failure, while there is no failure
- Repair cost - True Positives - Predicting failure correctly

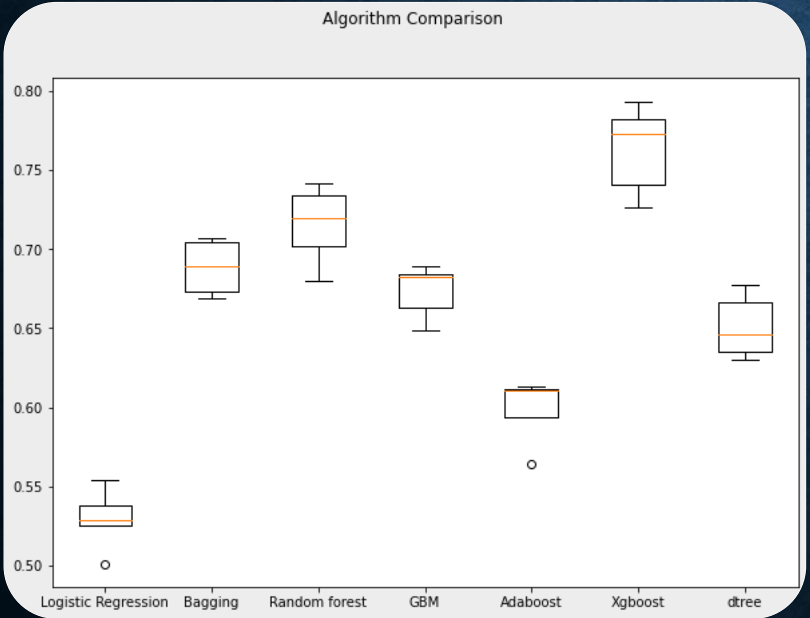
How to reduce the overall cost?

- 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

LOGISTIC, BAGGING, RANDOM FOREST, GBM, XGBOOST, ADABOOST, DECISION TREE MODELS ON TRAIN DATASET

- The different models were fitted on train data showed the following performance matrices:
- The models were tested using k-fold cross validation

- Minimum_Vs_Model cost is higher in training and very low in cross-validation. So all the models are overfitting on training data. The dataset has imbalance in the Target variable: 94.5% no-failure and only 5.5% failure. Lets perform oversampling and then check the results.

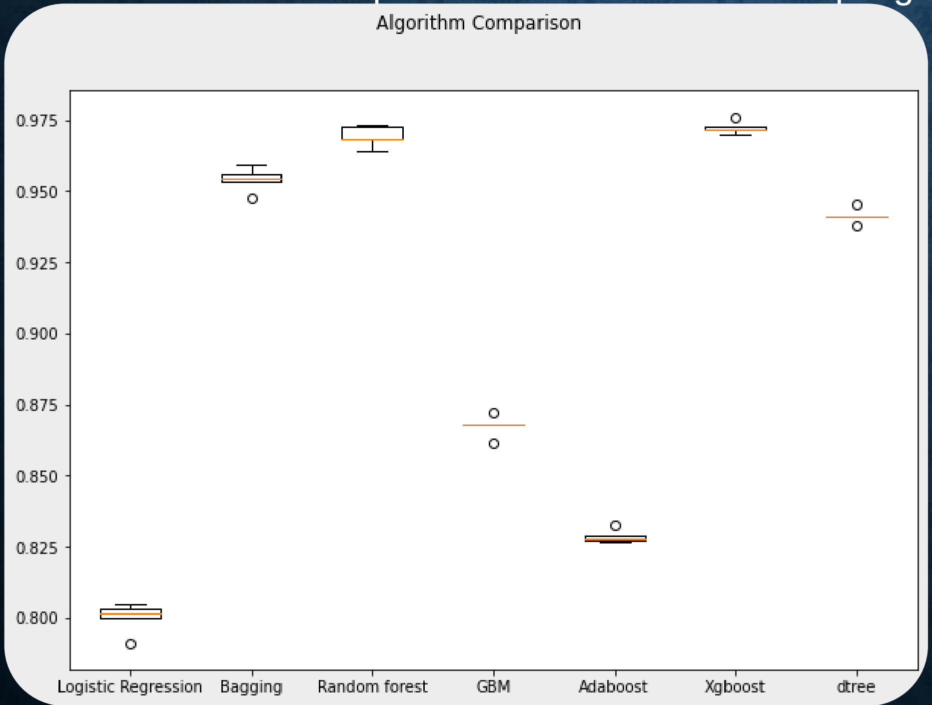



```
Cross-Validation Performance:

Logistic Regression: 52.948994378378856
Bagging: 68.86574469253958
Random forest: 71.52966747632996
GBM: 67.35879816374911
Adaboost: 59.88430481901255
Xgboost: 76.32838037155607
dtree: 65.12035300489133

Training Performance:

Logistic Regression:    Accuracy  Recall  Precision    F1  Minimum_Vs_Model_cost
0     0.967   0.485      0.853 0.619                  0.530
Bagging:    Accuracy  Recall  Precision    F1  Minimum_Vs_Model_cost
0     0.997   0.944      0.999 0.971                  0.914
Random forest:    Accuracy  Recall  Precision    F1  Minimum_Vs_Model_cost
0     1.000   1.000      1.000 1.000                  1.000
GBM:    Accuracy  Recall  Precision    F1  Minimum_Vs_Model_cost
0     0.987   0.780      0.981 0.869                  0.729
Adaboost:    Accuracy  Recall  Precision    F1  Minimum_Vs_Model_cost
0     0.976   0.637      0.890 0.743                  0.613
Xgboost:    Accuracy  Recall  Precision    F1  Minimum_Vs_Model_cost
0     1.000   1.000      1.000 1.000                  1.000
dtree:    Accuracy  Recall  Precision    F1  Minimum_Vs_Model_cost
0     1.000   1.000      1.000 1.000                  1.000
```

LOGISTIC, BAGGING, RANDOM FOREST, GBM, XGBOOST, ADABOOST, DECISION TREE MODELS AFTER SMOTE OVERSAMPLING ON TRAIN DATASET

- After UpSampling, the shape of train_X: (56720, 40) and the shape of train_y: (56720,)
- The different models were fitted on train data showed the following performance matrices:
- The models were tested using k-fold cross validation

In cross validation Minimum_Vs_Model cost is highest in XGBoost: 97.25 followed by Random Forest: 96.93 and Bagging: 95.41. Let’s check models performance with undersampling data.




```
Cross-Validation Performance:

Logistic Regression: 80.0088377103361
Bagging: 95.40942899036293
Random forest: 96.92752086101537
GBM: 86.76274431000867
Adaboost: 82.87444544309412
Xgboost: 97.25247518015713
dtree: 94.13857336531703

Training Performance:

Logistic Regression:
    Accuracy  Recall  Precision     F1  Minimum_Vs_Model_cost
0      0.874   0.876      0.874  0.875                  0.800
Bagging:
    Accuracy  Recall  Precision     F1  Minimum_Vs_Model_cost
0      0.999   0.998      1.000  0.999                  0.997
Random forest:
    Accuracy  Recall  Precision     F1  Minimum_Vs_Model_cost
0      1.000   1.000      1.000  1.000                  1.000
GBM:
    Accuracy  Recall  Precision     F1  Minimum_Vs_Model_cost
0      0.944   0.914      0.971  0.942                  0.868
Adaboost:
    Accuracy  Recall  Precision     F1  Minimum_Vs_Model_cost
0      0.905   0.894      0.914  0.904                  0.830
Xgboost:
    Accuracy  Recall  Precision     F1  Minimum_Vs_Model_cost
0      0.999   0.999      0.999  0.999                  0.998
dtree:
    Accuracy  Recall  Precision     F1  Minimum_Vs_Model_cost
0      1.000   1.000      1.000  1.000                  1.000
```

LOGISTIC, BAGGING, RANDOM FOREST, GBM, XGBOOST, ADABOOST, DECISION TREE MODELS AFTER UNDERSAMPLING ON TRAIN DATASET

- After DownSampling, the shape of train_X: (3280, 40) and the shape of train_y: (3280,)
- The different models were fitted on train data showed the following performance matrices:
- The models were tested using k-fold cross validation

All models in undersampled data have given worse result than oversampled data.

In undersampled data Random Forest is giving the highest score 84.7 and then the XGBoost: 84.04.

We can see that Random Forest is giving the highest cross-validated recall followed by XGBoost and then Gradient boosting classifier,

```
Cross-Validation Performance:

Logistic Regression: 77.65522788135344
Bagging: 81.69551921903387
Random forest: 84.67413072888996
GBM: 82.88956467253165
Adaboost: 80.05315674951575
Xgboost: 84.03896599877898
dtree: 77.45754336793459
```

```
Training Performance:

------------------------------------------------------------
Logistic Regression :
   Accuracy  Recall  Precision    F1  Minimum_Vs_Model_cost
0     0.859   0.855      0.862 0.859                  0.777
------------------------------------------------------------
Bagging :
   Accuracy  Recall  Precision    F1  Minimum_Vs_Model_cost
0     0.989   0.980      0.998 0.989                  0.967
------------------------------------------------------------
Random forest :
   Accuracy  Recall  Precision    F1  Minimum_Vs_Model_cost
0     1.000   1.000      1.000 1.000                  1.000
------------------------------------------------------------
GBM :
   Accuracy  Recall  Precision    F1  Minimum_Vs_Model_cost
0     0.952   0.918      0.985 0.950                  0.877
------------------------------------------------------------
Adaboost :
   Accuracy  Recall  Precision    F1  Minimum_Vs_Model_cost
0     0.906   0.894      0.916 0.905                  0.831
------------------------------------------------------------
Xgboost :
   Accuracy  Recall  Precision    F1  Minimum_Vs_Model_cost
0     1.000   1.000      1.000 1.000                  1.000
------------------------------------------------------------
dtree :
   Accuracy  Recall  Precision    F1  Minimum_Vs_Model_cost
0     1.000   1.000      1.000 1.000                  1.000
```

XGBOOST AFTER GRID SEARCH HYPERTUNING BEST PARAMETERS

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', 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, random_state=1,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=10, subsample=0.9,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

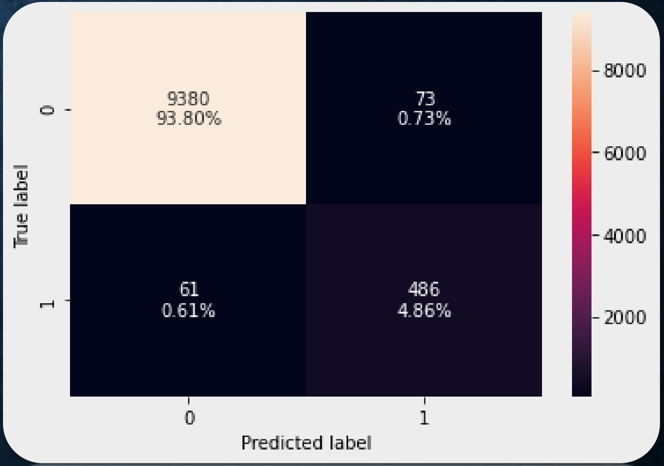
Training performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	1.000	1.000	1.000	1.000	1.000

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.987	0.888	0.869	0.879	0.813

- XGBoost is giving good Minimum_Vs_Model_cost in validation set. Although its overfitting



XGBOOST AFTER RANDOMIZED SEARCH HYPERTUNING BEST PARAMETERS

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.2, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=200, n_jobs=8, num_parallel_tree=1, random_state=1,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=10, subsample=0.9,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

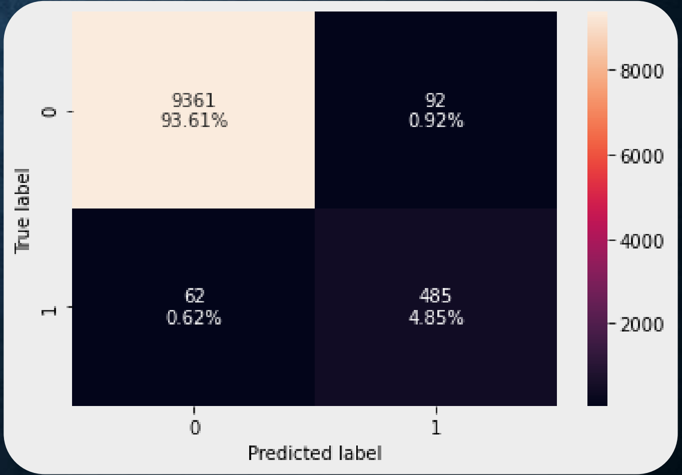
Training performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.999	1.000	0.999	0.999	1.000

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.985	0.887	0.841	0.863	0.803

- XGBoost is giving good Minimum_Vs_Model_cost in validation set. Although its overfitting



RANDOM FOREST AFTER GRID SEARCH HYPERTUNING BEST PARAMETERS

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

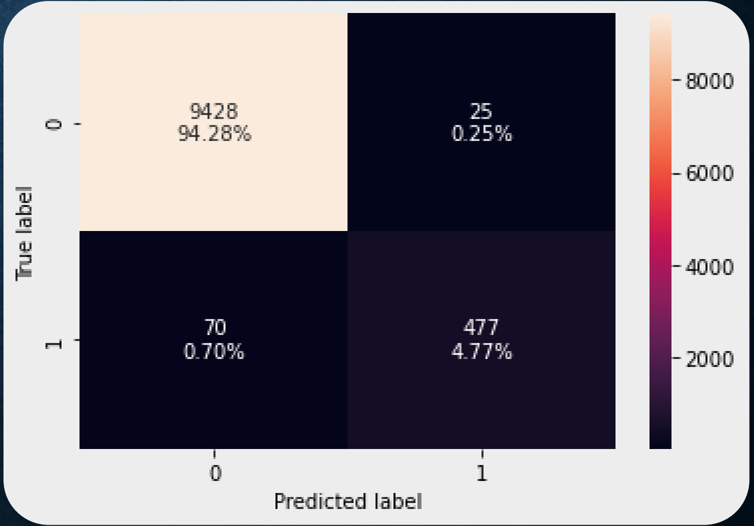
Training performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.999	0.998	1.000	0.999	0.997

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.991	0.872	0.950	0.909	0.814

- Random Foirest is also giving good Minimum_Vs_Model_cost in validation set. Although its overfitting



RANDOM FOREST AFTER RANDOMIZED SEARCH HYPERTUNING BEST PARAMETERS

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

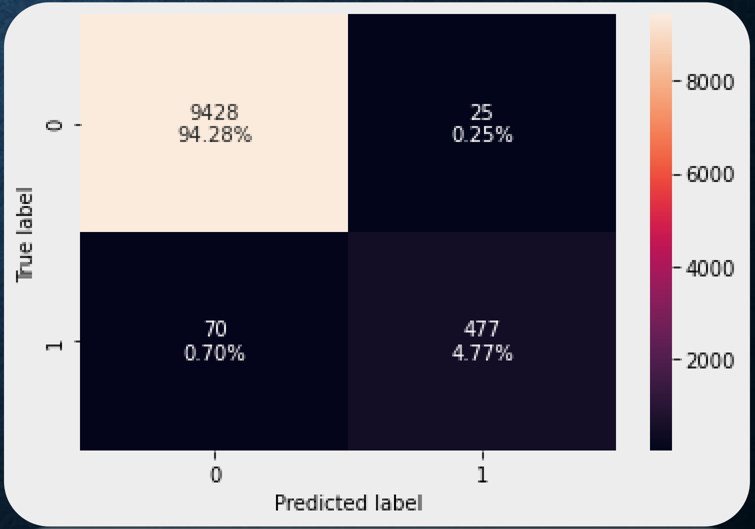
Training performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.999	0.998	1.000	0.999	0.997

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.991	0.872	0.950	0.909	0.814

- Random Forest is also giving good Minimum_Vs_Model_cost in validation set. Although its overfitting



BAGGING AFTER GRID SEARCH HYPERTUNING BEST PARAMETERS

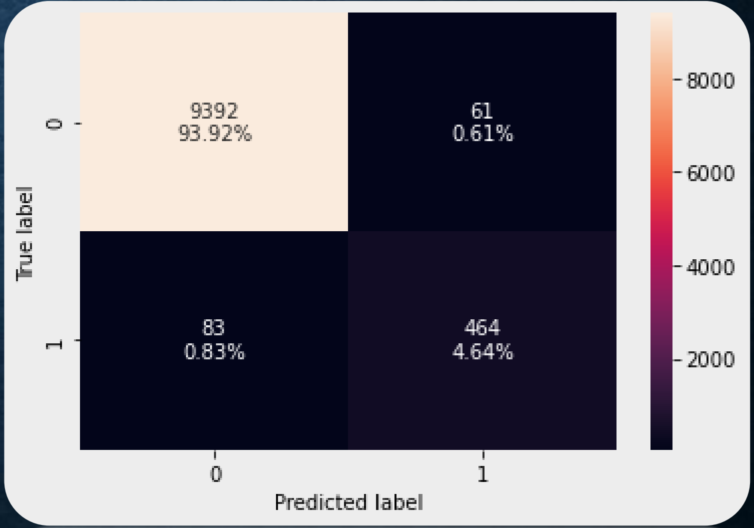
```
BaggingClassifier(max_features=0.9, max_samples=0.9, n_estimators=50,
                  random_state=1)
```

Training performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	1.000	1.000	1.000	1.000	1.000

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.986	0.848	0.884	0.866	0.775



- Bagging is just giving barely acceptable Minimum_Vs_Model_cost in validation set. Although its overfitting

BAGGING AFTER RANDOMIZED SEARCH HYPERTUNING BEST PARAMETERS

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

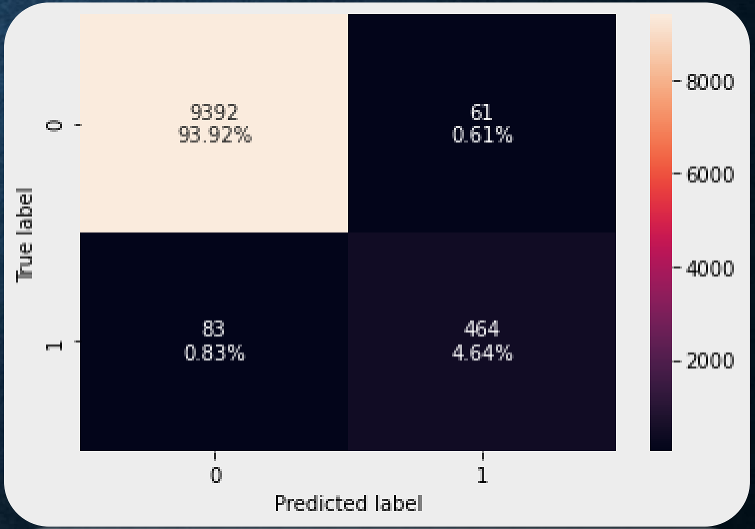
Training performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	1.000	1.000	1.000	1.000	1.000

Validation performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.986	0.848	0.884	0.866	0.775

- Bagging is just giving barely acceptable Minimum_Vs_Model_cost in validation set. Although its overfitting



MODEL PERFORMANCE COMPARISON AND CONCLUSIONS

- Training performance of tuned models

	Xgboost Tuned with Grid search	Xgboost Tuned with Random Search	Random Forest Tuned with Grid search	Random Forest Tuned with Random search	Bagging Classifier Tuned with Grid search	Bagging Classifier Tuned with Random search
Accuracy	1.000	0.999	0.999	0.999	1.000	1.000
Recall	1.000	1.000	0.998	0.998	1.000	1.000
Precision	1.000	0.999	1.000	1.000	1.000	1.000
F1	1.000	0.999	0.999	0.999	1.000	1.000
Minimum_Vs_Model_cost	1.000	1.000	0.997	0.997	1.000	1.000

Minimum_Vs_Model_cost is 1 in both XGBoost and Bagging classifier and 0.999 in Random forest classifier.

```
Validation performance comparison:
```

	Xgboost Tuned with Grid search	Xgboost Tuned with Random Search	Random Forest Tuned with Grid Search	Random Forest Tuned with Random Search	Bagging Classifier Tuned with Grid Search	Bagging Classifier Tuned with Random Search
Accuracy	0.987	0.985	0.991	0.991	0.986	0.986
Recall	0.888	0.887	0.872	0.872	0.848	0.848
Precision	0.869	0.841	0.950	0.950	0.884	0.884
F1	0.879	0.863	0.909	0.909	0.866	0.866
Minimum_Vs_Model_cost	0.813	0.803	0.814	0.814	0.775	0.775

- Both XGBoost tuned and Random Forest tuned are giving generalised performance.
- XGBoost grid: 0.813, XGBoost random: 0.803,
- Random Forest grid: 0.814, Random Forest Random: 0.814
- I'll choose the Random Forest random search as it's execution time is comparatively less and it is giving a slightly better performance than XGBoost

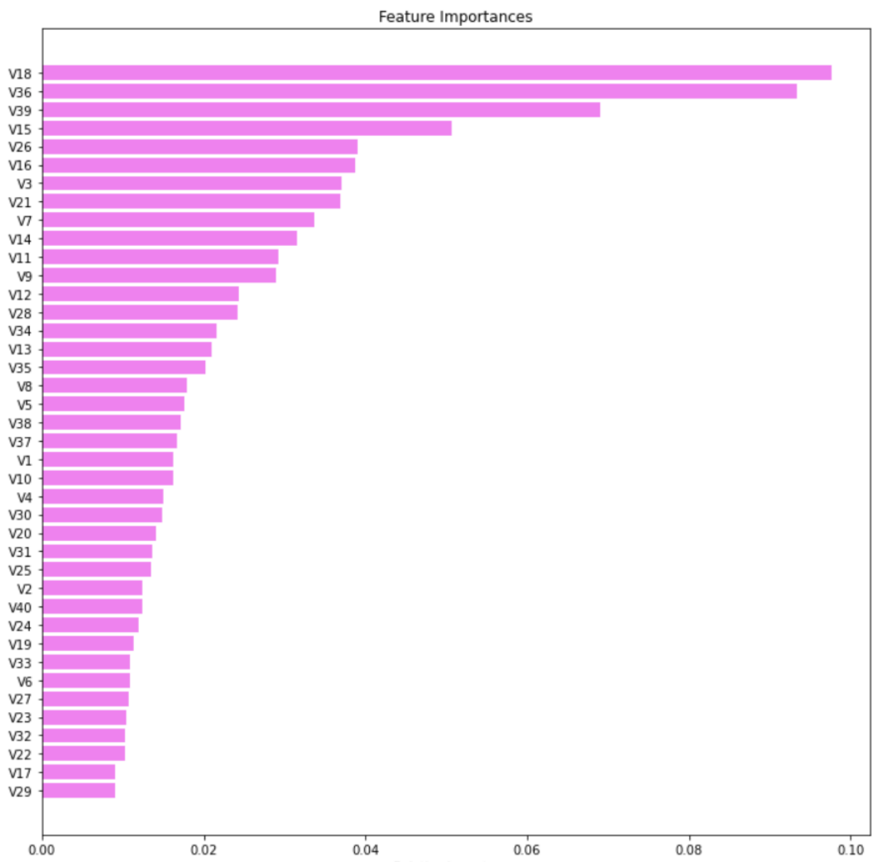
FINAL MODEL:TUNED RANDOM FOREST CLASSIFIER

The Tuned Random Forest Classifier model was giving a much better performance

Test performance:

	Accuracy	Recall	Precision	F1	Minimum_Vs_Model_cost
0	0.990	0.857	0.951	0.902	0.799

The 10 most important features out of 40 are V18, V36, V39, V15, V26, V16, V3, V21, V7 and V14.



PIPELINE USING THE RANDOM FOREST CLASSIFIER

```
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', ...])])),
                ('Random Forest Classifier',
                 RandomForestClassifier(max_features='sqrt',
                                        max_samples=0.5000000000000001,
                                        n_estimators=250, random_state=1))])
```

ACTIONABLE INSIGHTS AND RECOMMENDATIONS

Conclusion

The 10 most important features out of 40 are V18, V36, V39, V15, V26, V16, V3, V21, V7 and V14.

The input from these features should be considered when making predictions.

THE END

By: syeda ambreen karim bukhari