ReneWind- Model Tuning Project

UT DSBA

June 2023

Business Problem Overview and Solution Approach

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.).

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, 20000 observations in the training set and 5000 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 generators could be repaired before failing/breaking to reduce the overall maintenance cost.

Objective (cont.)

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

Data Description

- - The data provided is a transformed version of original data which was collected using sensors.
- - Train.csv - To be used for training and tuning of models.
- - Test.csv - To be used only for testing the performance of the final best model.
- - Both the datasets consist of 40 predictor variables and 1 target variable

Data Overview

- The training data contains 20,000 rows with 41 columns
- The test data contains 5,000 rows with 41 columns
- The 40 of the columns are float types with one type being an integer.
- There are no object columns.
- There are no duplicate values in the data.
- There are 18 missing values in both row V1 and V2..

EDA Results

Histograms

We can see that distribution of most features is approximately normal.

- The distribution of very few features is skewed.
- The average and median values for most features are close to zero.

Link to Appendix slide on data background check

EDA

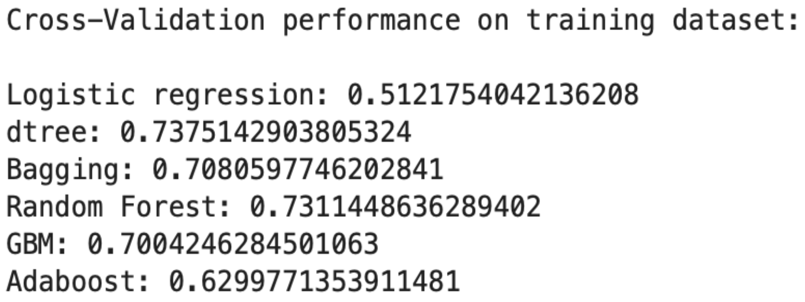
The data set has already been split into train and test sets.

The train set then has been split into train and validation sets which both contain 6,000 rows and 40 columns.

The initial test set has been split into X test and y test which contain 5,000 rows and 40 columns of data.

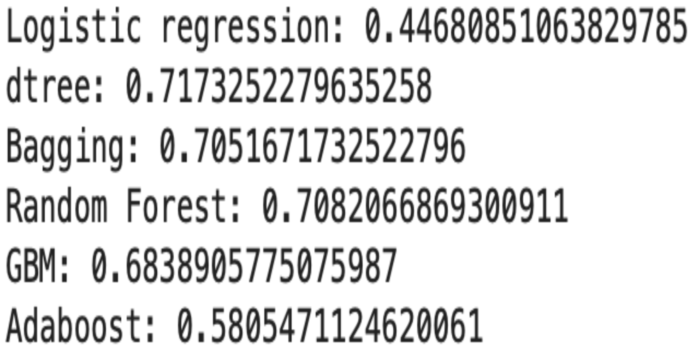
The columns were imputed with the median for any missing values to prevent data leakage. There are no missing values in the new sets.

Model Building On Original Data



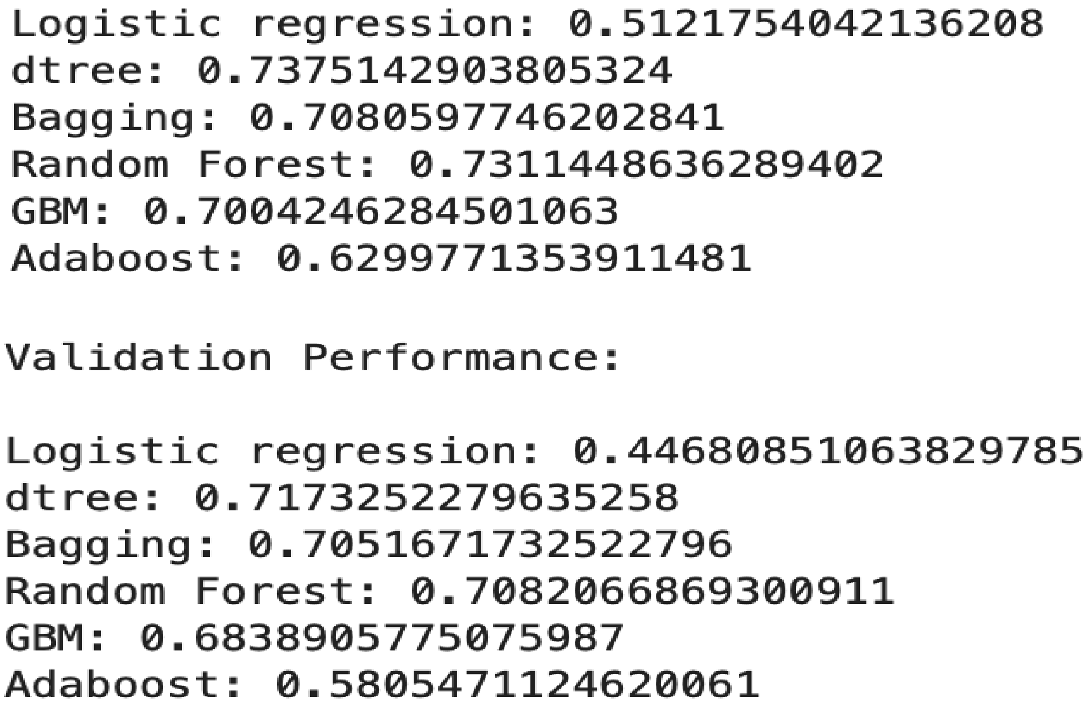
```
Cross-Validation performance on training dataset:

Logistic regression: 0.5121754042136208
dtree: 0.7375142903805324
Bagging: 0.7080597746202841
Random Forest: 0.7311448636289402
GBM: 0.7004246284501063
Adaboost: 0.6299771353911481
```

Validation Performance:



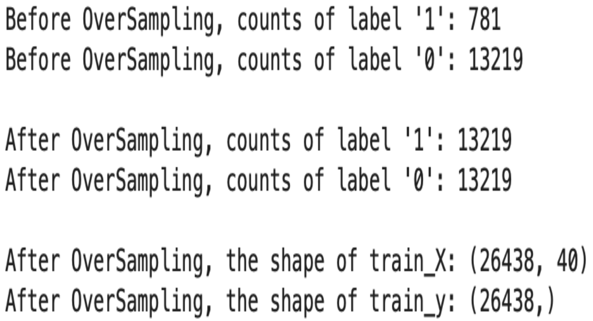
```
Logistic regression: 0.44680851063829785
dtree: 0.7173252279635258
Bagging: 0.7051671732522796
Random Forest: 0.7082066869300911
GBM: 0.6838905775075987
Adaboost: 0.5805471124620061
```

Cross-validation performance on training set

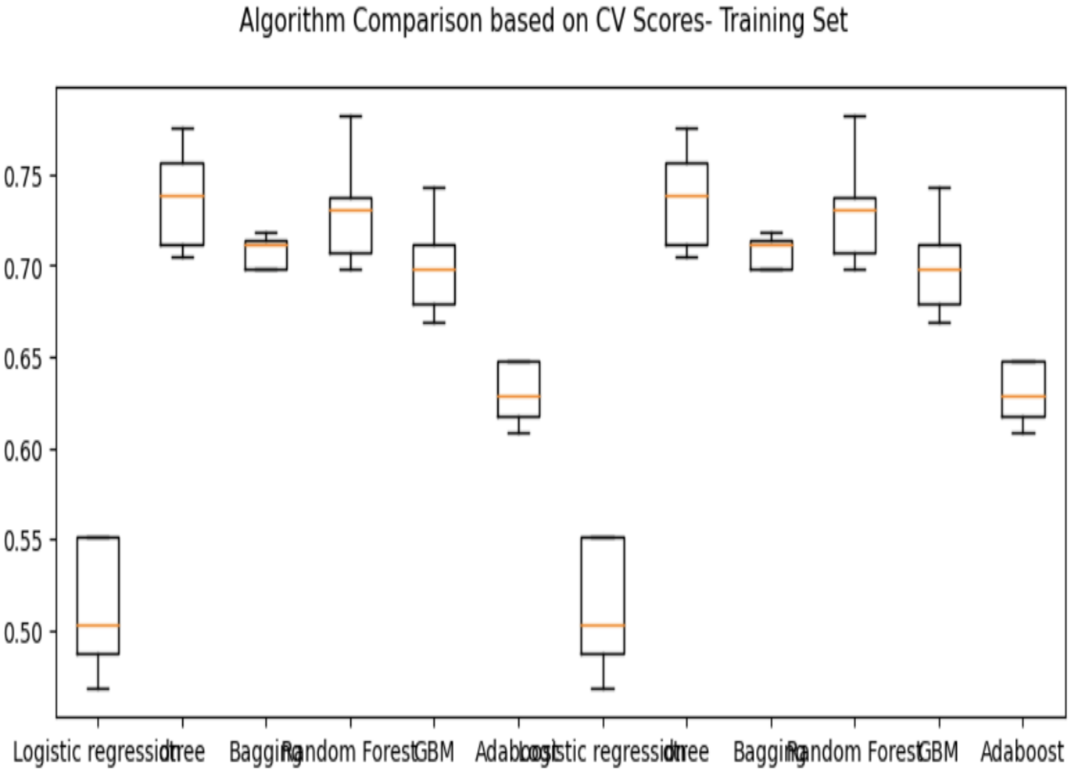


```
Logistic regression: 0.5121754042136208
dtree: 0.7375142903805324
Bagging: 0.7080597746202841
Random Forest: 0.7311448636289402
GBM: 0.7004246284501063
Adaboost: 0.6299771353911481

Validation Performance:

Logistic regression: 0.44680851063829785
dtree: 0.7173252279635258
Bagging: 0.7051671732522796
Random Forest: 0.7082066869300911
GBM: 0.6838905775075987
Adaboost: 0.5805471124620061
```

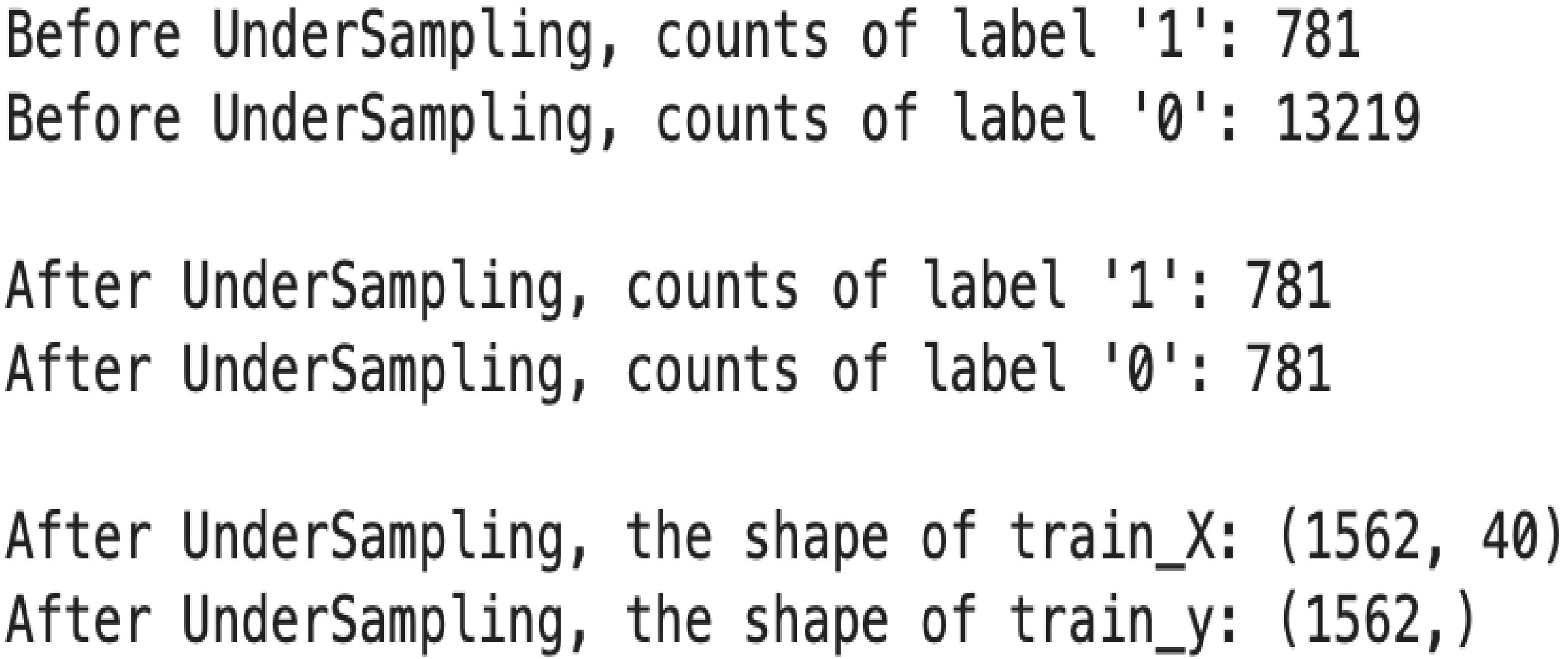
Model Building with oversampled Data



```
Before OverSampling, counts of label '1': 781
Before OverSampling, counts of label '0': 13219

After OverSampling, counts of label '1': 13219
After OverSampling, counts of label '0': 13219

After OverSampling, the shape of train_X: (26438, 40)
After OverSampling, the shape of train_y: (26438,)
```



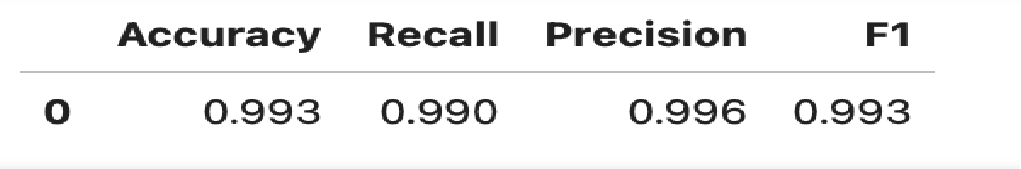
Model Building with Undersampled Data



```
Before UnderSampling, counts of label '1': 781
Before UnderSampling, counts of label '0': 13219

After UnderSampling, counts of label '1': 781
After UnderSampling, counts of label '0': 781

After UnderSampling, the shape of train_X: (1562, 40)
After UnderSampling, the shape of train_y: (1562,)
```

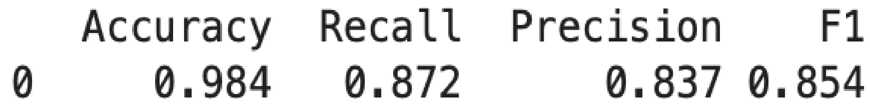
Hyperparameter Tuning

- Tuning AdaBoost using oversampled data
- Ada Boost train performance



	Accuracy	Recall	Precision	F1
0	0.993	0.990	0.996	0.993

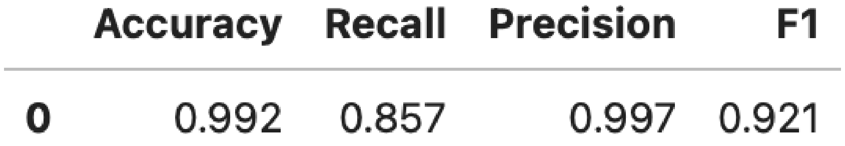
Ada Validation performance



```
   Accuracy  Recall  Precision    F1
0     0.984   0.872      0.837 0.854
```

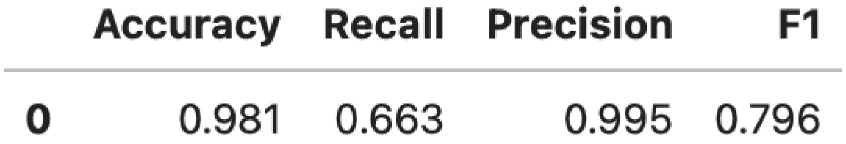
Random Forest Using undersampled data

Train performance



	Accuracy	Recall	Precision	F1
0	0.992	0.857	0.997	0.921

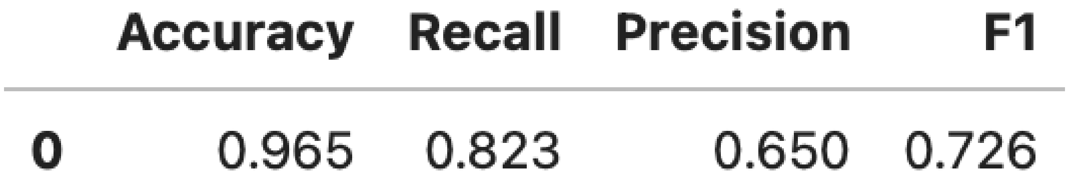
Validation performance



	Accuracy	Recall	Precision	F1
0	0.981	0.663	0.995	0.796

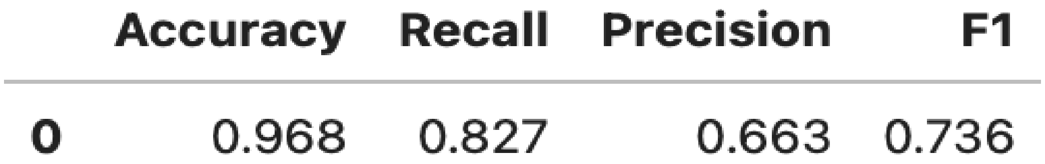
Gradient Boosting using oversampled Data

Training performance



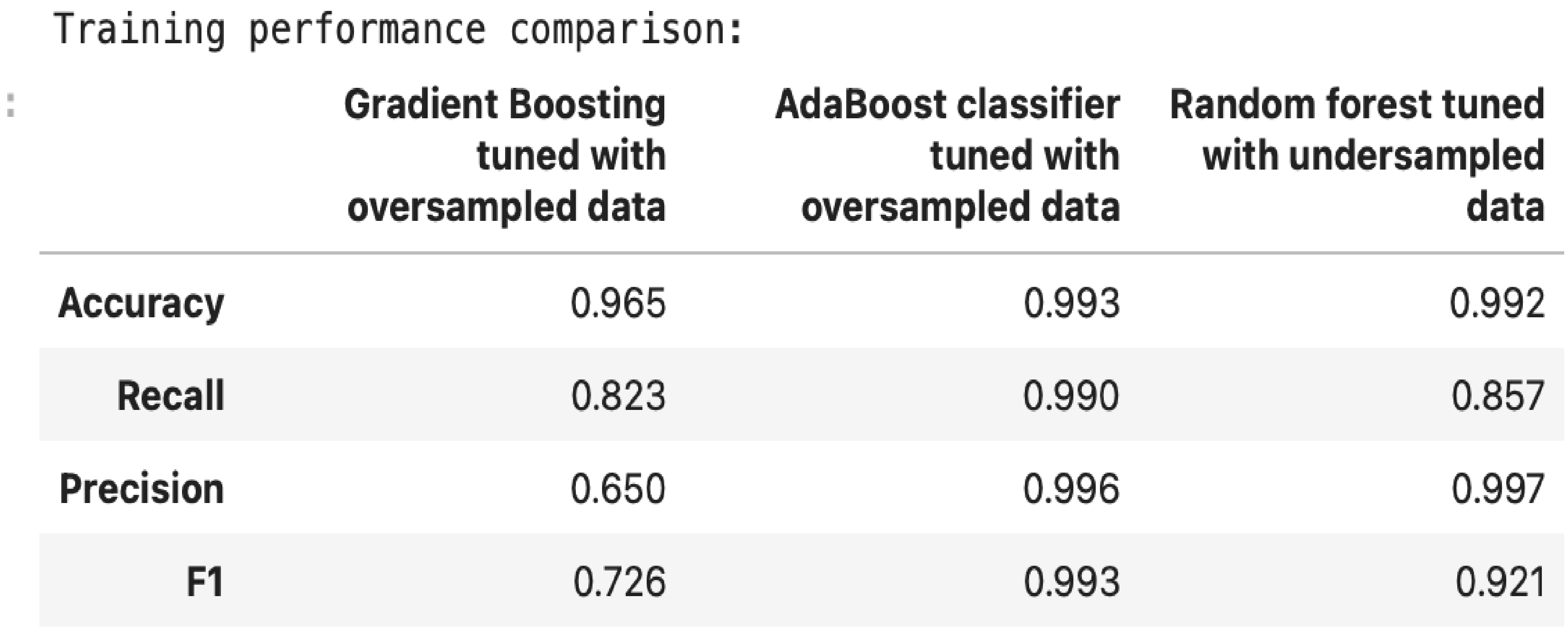
	Accuracy	Recall	Precision	F1
0	0.965	0.823	0.650	0.726

Validation performance



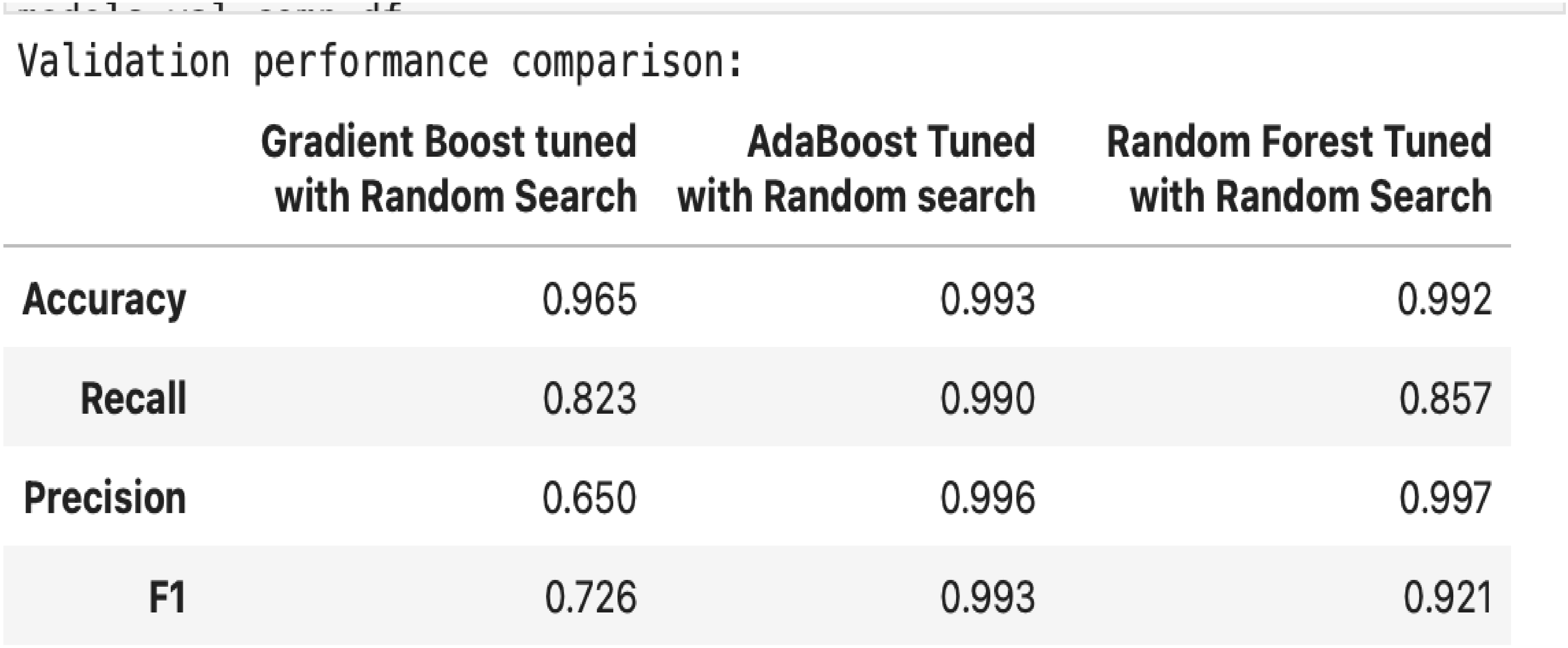
	Accuracy	Recall	Precision	F1
0	0.968	0.827	0.663	0.736

Model comparison and choosing Final model



```
Training performance comparison:
```

	Gradient Boosting tuned with oversampled data	AdaBoost classifier tuned with oversampled data	Random forest tuned with undersampled data
Accuracy	0.965	0.993	0.992
Recall	0.823	0.990	0.857
Precision	0.650	0.996	0.997
F1	0.726	0.993	0.921



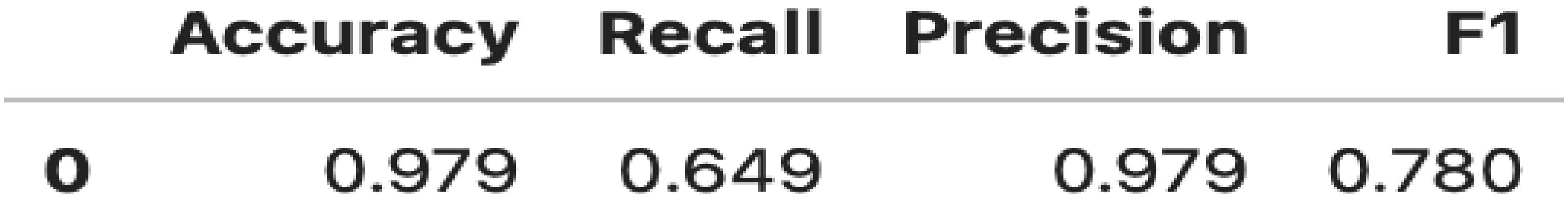
```
models_val_comp_df
```

```
Validation performance comparison:
```

	Gradient Boost tuned with Random Search	AdaBoost Tuned with Random search	Random Forest Tuned with Random Search
Accuracy	0.965	0.993	0.992
Recall	0.823	0.990	0.857
Precision	0.650	0.996	0.997
F1	0.726	0.993	0.921

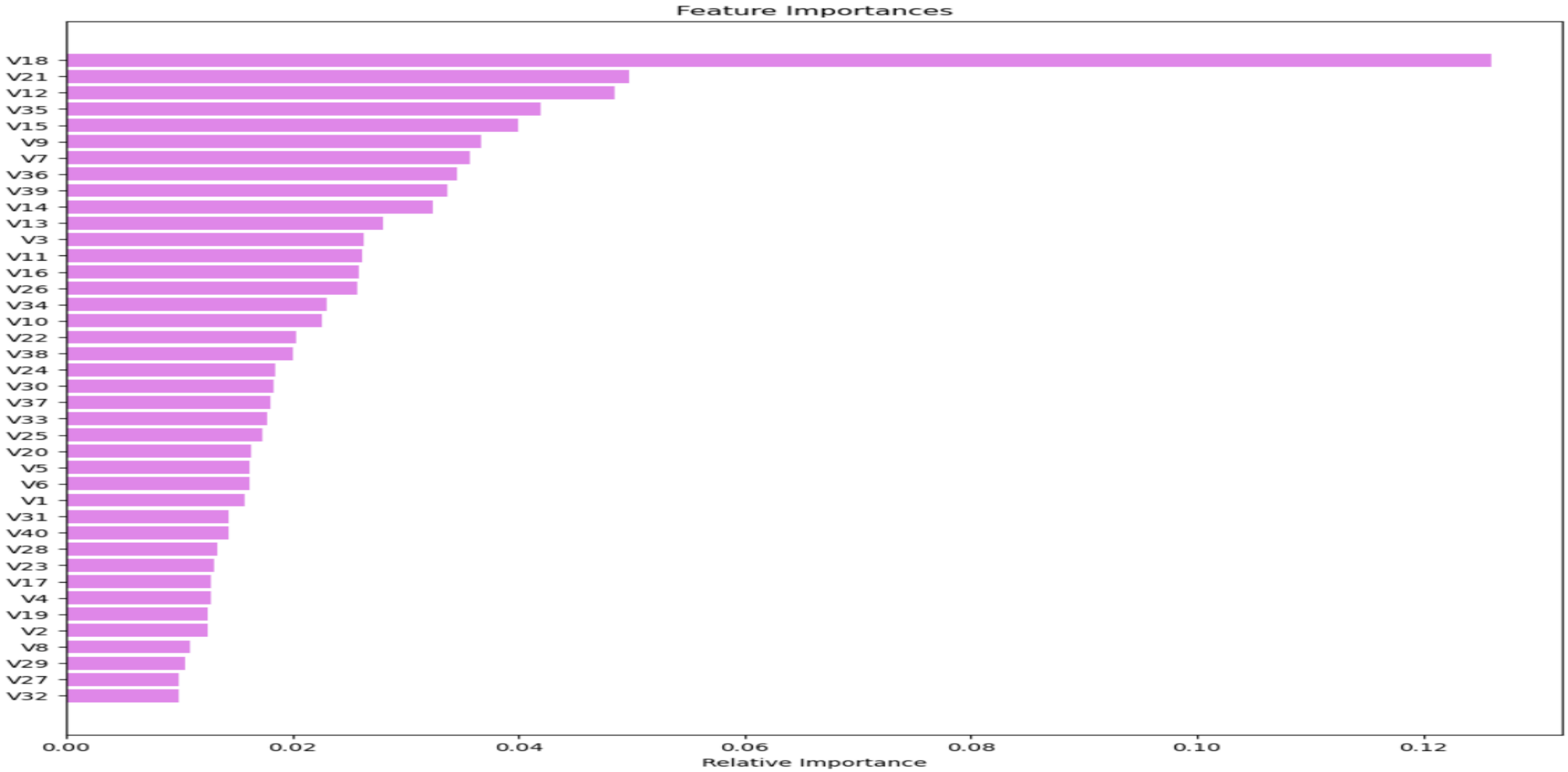
Choice of final model

- Performance of final model on test



	Accuracy	Recall	Precision	F1
0	0.979	0.649	0.979	0.780

Feature Importances



Great Learning

Happy Learning !