

# Machine Learning Engineer Nanodegree

## Capstone

### Project: Air pressure system failures in Scania trucks

Hajime Kawata

February 26st, 2019

Predict failure of heavy duty trucks out of sensors data

#### Domain Background

Reducing the failure rate of trucks in logistics is an important issue for improving profitability and customer satisfaction in logistics business. Until now, logistics companies rely on regular maintenance to prevent the occurrence of failures. However, since the utilization rate of trucks exceeds the days what has been assumed to be covered by periodic maintenance, the percentage of post-maintenance caused by failures increasing, which has been a problem for some time.

For this reason, it is strongly required that inspection and maintenance work be performed before regular failure, by appropriately detecting signs of failure during regular maintenance In order to properly capture signs of failure, it has been considered to use data collected from sensors installed in trucks, bu sensor data is gathered in milliseconds at longest, and collection targets are also diverse. As a result, several hundred megabytes of data may be generated from one truck on a day. A large data cost has been anticipated.

In this project in order to quickly and flexibly arrange trucks based on failure prediction,

- Reduce the amount of features derived from the target sensor to be collected and suppress the arithmetic cost to be generated
- Perform a failure based on the logic derived from the data pattern leading to the failure on the truck side.

This would make it possible to appropriately send alerts before the actual failure occurrence to the monitoring center and to adjust the arrangement of the trucks.

#### Problem Statement

Here, using the training data, we derive the prediction model of the occurrence of the failure (label) from the sensor data (features). This prediction logic is applied to the test data and the ability of the prediction logic is evaluated. The sensor data is anonymized for confidentiality reasons, and so we have to make the prediction model purely on mathematical and statistical approach.

#### Solution Statement

##### Project Design

1. Split the training data set to establish model and evaluate
2. Apply PCA to reduce the dimension of the model.
3. Device predictor with XGBoost classifier, while seeking the best hyperparameter values with Grid Search by ROC-AUC.
4. Evaluate with the model with given test set, with confusion matrix, and accuracy, precision, and f1 scores.
5. Using the given test set to evaluate the model
6. Evaluate the contribution of feature, to discuss the possibility of reducing the sensors to predict failures.

##### XGBoost

本プロジェクトでは、対象となるAir pressure systemを高い精度で検出するための機械学習の手法として、XGBoost (<https://xgboost.readthedocs.io/en/latest/index.html>)によるアプローチを見る。多様な問題に対し、良い結果を出すことが知られている。Boosted treesは、アンサンブル学習と呼ばれる手法の一つで、Gradient Boostと呼ばれる学習を加速する手法と、Random Forestと呼ばれるベストなモデルを構築する方法とを組み合わせた手法です。Random Forestで発生するOver Fittingを予防するRegularization Modelが組み込まれている。

Boosted treesの予測精度はRandom Forestsよりも向上しますが、チューニングが必要なパラメータが複数存在する。ここではこのパラメータの調整に、Grid Searchと呼ばれる手法を用いる。汎化能力を上げるために、学習率(XGBoostパッケージではパラメータeta)を下げていき、その都度最適化を行う

複数のパラメータ調整によるチューニングが必要であり、最適化はグリッドサーチやCross Validationを複数行う

It is computationally demanding to enumerate all the possible splits for continuous features. In order to do so efficiently, the algorithm must first sort the data according to feature values and visit the data in sorted order to accumulate the gradient statistics for the structure score  $i$

データの特徴について分析し、モデル構築の際のデータ前処理の要否、前処理の内容について検討する。

For the criteria listed in **Domain Background\***, take the following solution approach in this project

- Reduce sensor data features from 171 by PCA (Primary Components Analysis)
- Construct the model from reduced features with XGBoost classifier, by adjusting hyperparameters with Grid Search CV

#### Data Exploration

In this section, you will begin exploring the data through visualizations and code to understand how each feature is related to the others. You will observe a statistical description of the dataset, consider the relevance of each feature, and select a few sample data points from the dataset which you will track through the course of this project.

このデータ・セットにおいて訓練データとテストデータは所与のものとして指定されているので、これらを用いる。センサーデータは8bit単位でセンサーから取得される（参照 I2CやSPIなどの電子回路では1byte(=8bit)データとして取得できる）。これについては、実際の物理的な計測値に変換された下記データを用いるものとする。`train = pd.read_csv('data/aps_failure_training_set_processed_8bit.csv')` `test = pd.read_csv('data/aps_failure_test_set_processed_8bit.csv')`

## Datasets and Inputs

Data is provided in : Kaggle : <https://www.kaggle.com/uciml/aps-failure-at-scania-trucks-data-set/home>

The dataset consists of data collected from heavy Scania trucks in everyday usage. The system in focus is the Air Pressure system (APS) which generates pressurized air that is utilized in various functions in a truck, such as braking and gear changes. (Kaggle)

このデータ・セットでは以下の2つのクラスに分類されたデータが与えられている。

Category	Description
positive class	consists of component failures for a specific component of the APS system.
negative class	consists of trucks with failures for components not related to the APS.

Following is the data volume information.

	Total	Positive	Negative	features
Train	60000	1000	59000	171
Test	16000	-	-	171

Positive Classになるデータ・セットを適切に分類することが求められる。以下で、対象となる2つのクラスは以下の値で識別されていることから、positive class 0.992188 negative class -0.992188 positive classを1, negative classを0で新たにラベル付し直す。

そのため、このプロジェクトではポジティブクラスになるデータセット、つまりクラスが1となるデータ・セットを適切に分類する、予測モデルを構築する。

```
train = pd.read_csv('data/aps_failure_training_set_processed_8bit.csv')
test = pd.read_csv('data/aps_failure_test_set_processed_8bit.csv')
```

display(train.shape) (60000, 171) display(test.shape) (16000, 171)

trainデータ・セットの例

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	class	aa_000	ab_000	ac_000	ad_000	ae_000	af_000	ag_000	ag_001	ag_002	...	ee_002	ee_003
0	-0.992188	0.117188	-0.289062	0.992188	-0.007812	-0.046875	-0.054688	-0.007812	-0.03125	-0.054688	...	0.687500	0.515625
1	-0.992188	-0.179688	-0.289062	-0.468750	-0.007812	-0.046875	-0.054688	-0.007812	-0.03125	-0.054688	...	-0.023438	-0.062500
2	-0.992188	-0.125000	-0.289062	-0.468750	-0.007812	-0.046875	-0.054688	-0.007812	-0.03125	-0.054688	...	-0.140625	-0.093750
3	-0.992188	-0.406250	-0.289062	-0.468750	-0.007812	-0.046875	-0.007812	-0.007812	-0.03125	-0.054688	...	-0.382812	-0.382812
4	-0.992188	0.007812	-0.289062	-0.468750	-0.007812	-0.046875	-0.054688	-0.007812	-0.03125	-0.054688	...	0.156250	0.031250

5 rows × 171 columns

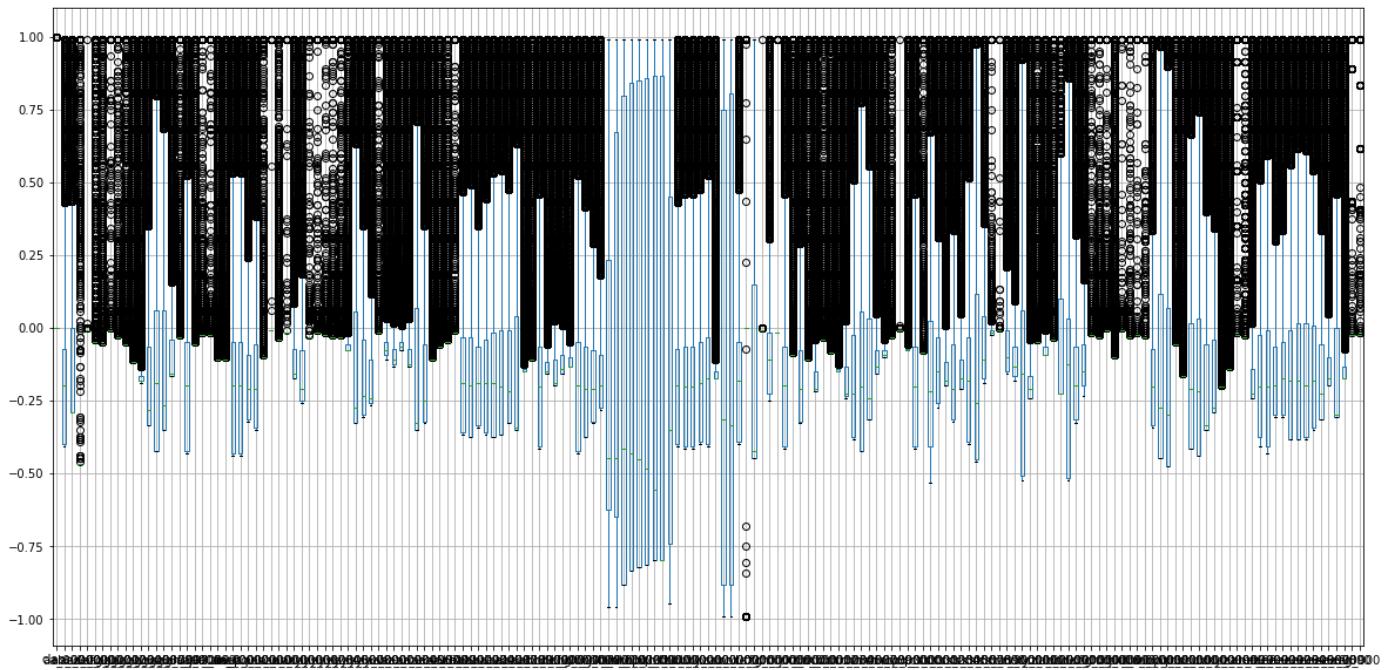
データに対し必要な特徴量の前処理について検討する。特徴量ごとの値のスケールが異なる場合、計算機による計算誤差が、予測に影響を及ぼす恐れがある。また、極端な偏りがある場合、予測精度に影響を及ぼす場合がある。予測モデルを構築するにあたり、所与のトレーニングデータの特徴を把握したうえで必要な前処理を行い、これらの影響を極力排除するのが慣例である。

## データの広がりとスケールの調整の要否

まず、データの偏りを見るにあたり、171個の各特徴量をボックス図にプロットした。全てのデータが-1から1の範囲に収まっている。75パーセンタイルについては、その幅は特徴量ごとにまちまちである。-1から1までの範囲に広がっているもの、0.01のオーダーに収まっているものなどがある。

また、Outlierと呼ばれる75パーセンタイルに収まらないデータ（図中の黒い点）が、各特徴量において広がりを持っていることが見られる。

以下では、上記の分布の偏りや、Outlierについて特徴を分析し、処理すべきか否か、処理すべきであるならばどのような処理を行うべきか検証する。



具体的に、一部のデータのパーセンタイル値をみると、25, 50, 75の各パーセンタイルで同じ値を持つセンサーデータ値が多い傾向にあることが見て取れる。極端に一部の値に集中していることがわかる。

これは、対象がセンサーによるものであることから、多くは故障状態以外の定常状態を計測しているとすれば、十分ありうる状況である。機械の異常診断を行う際には、圧倒的多数の正常データから、少ない異常データを導くモデルを構築する必要があると考えられる。

訓練データセット中の特徴量の各統計データの一部を表示する。

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	class	aa_000	ab_000	ac_000	ad_000	ae_000	af_000	ag_000	ag_001
count	60000.000000	60000.000000	60000.000000	60000.000000	60000.000000	60000.000000	60000.000000	60000.000000	60000.000000
mean	0.016667	-0.124611	-0.071121	-0.198529	-0.007737	-0.033483	-0.040633	-0.006584	-0.026241
std	0.128020	0.367680	0.356812	0.564872	0.004138	0.107086	0.111752	0.032016	0.065200
min	0.000000	-0.406250	-0.289062	-0.468750	-0.007812	-0.046875	-0.054688	-0.007812	-0.031250
25%	0.000000	-0.398438	-0.289062	-0.468750	-0.007812	-0.046875	-0.054688	-0.007812	-0.031250
50%	0.000000	-0.195312	-0.289062	-0.468750	-0.007812	-0.046875	-0.054688	-0.007812	-0.031250
75%	0.000000	-0.070312	-0.000000	-0.468750	-0.007812	-0.046875	-0.054688	-0.007812	-0.031250
max	1.000000	0.992188	0.992188	0.992188	0.992188	0.992188	0.992188	0.992188	0.992188

8 rows × 171 columns

APSの異常を示すクラス1の実際の割合を見ると、訓練データは全部で60,000件あり、そのうちの1,000件（17%）である。これは、171の多くのセンサーは、APSの異常発生時においても、定常状態と同じ値を異常時も示している可能性がある。171の特徴量の中から、異常状態を判別するための組み合わせを人出で行うのは、非現実的であり、科学的なアプローチを通じて自動的に行う方法を提案する必要がある。

y.value\_counts()

0	59000
1	1000

約1.7%のデータがAPS関連の故障データである。

テストデータについては 0 15625 1 375

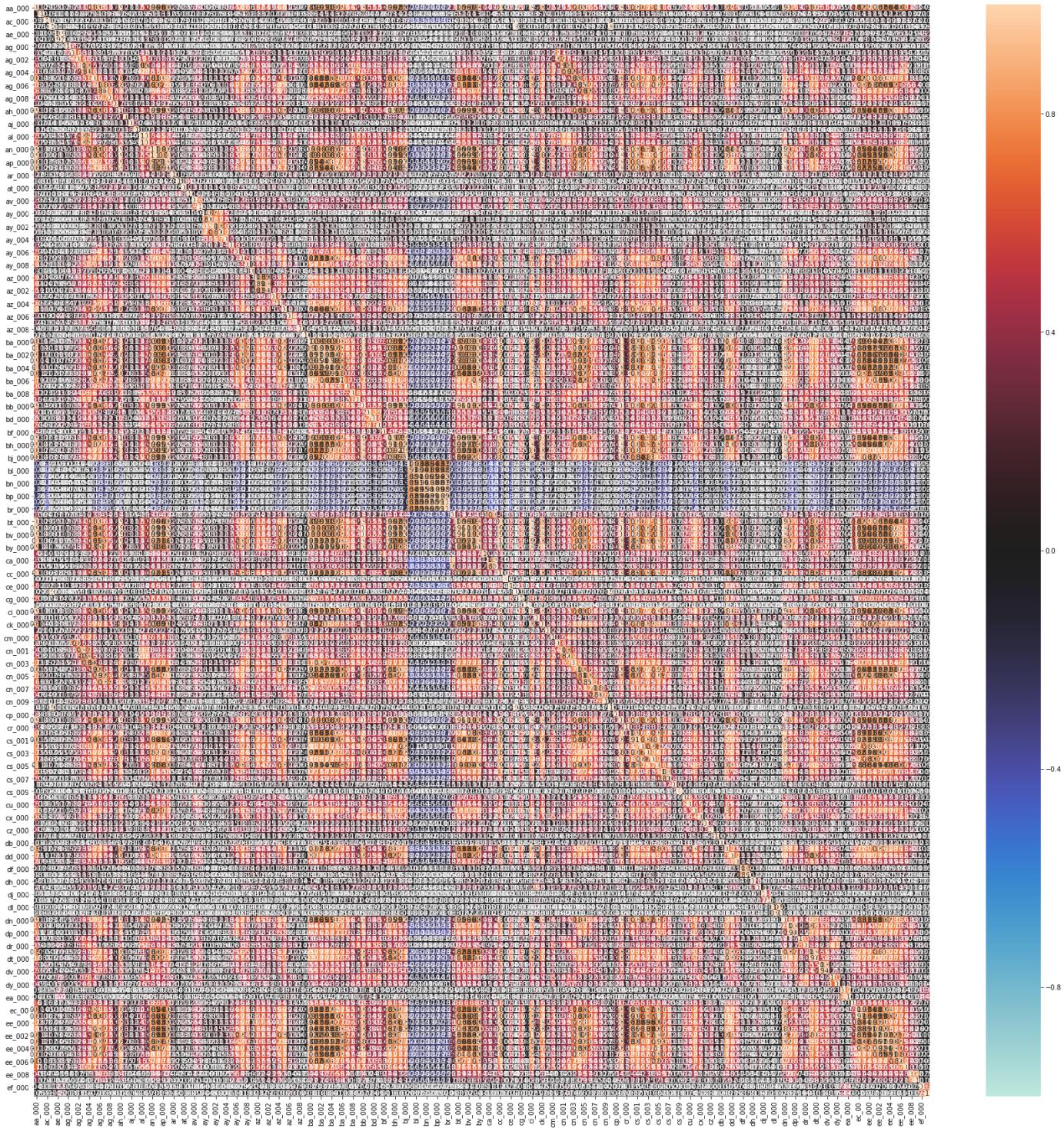
約2.3%のデータがAPS関連の故障データである。 0.0234375

前処理を検討するにあたり、次に、今回採用したデータの特徴量間の相関について調査する。

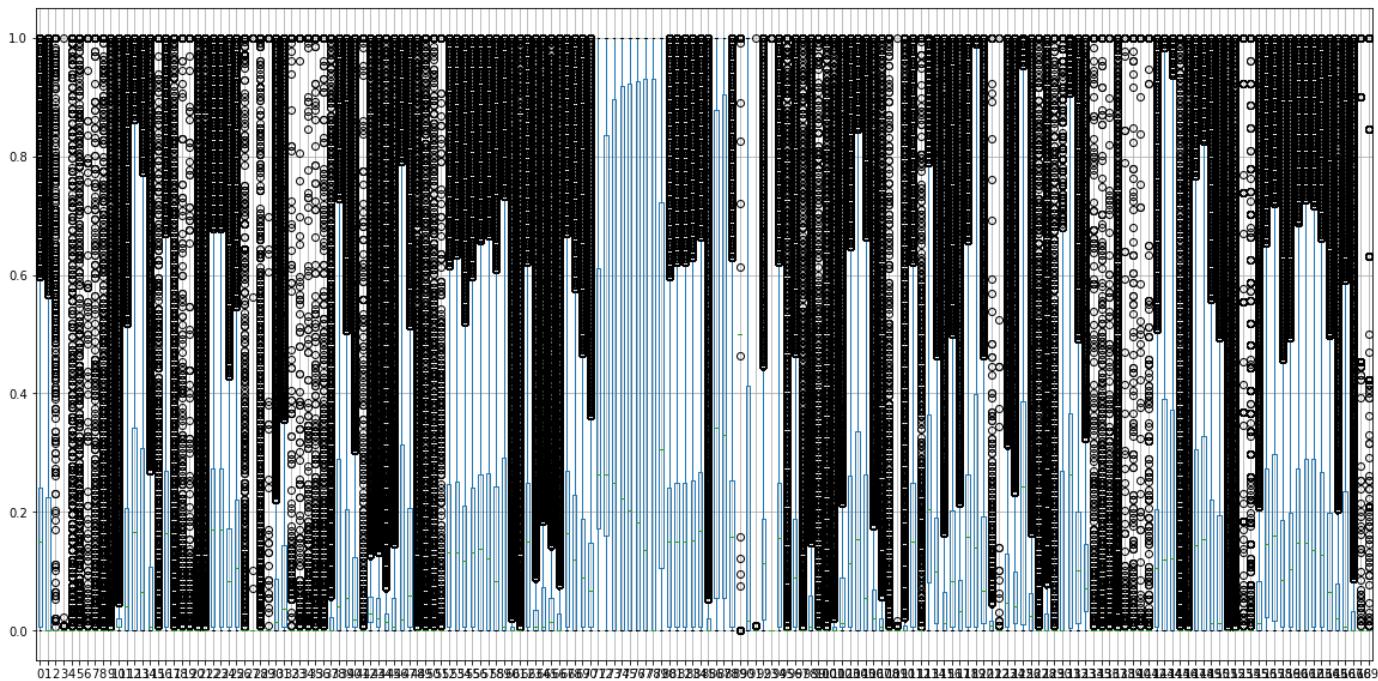
センサーデータ間の相関係数をヒートマップとして図示した。このヒートマップは正の相関が強いほど白に近い赤、弱いほど白に近い青で表示される。相関の強さとヒートマップの色との関係については、図の左の棒に尺度が示されている。ヒートマップ中に白に近い赤となっている特徴量の組み合わせが全体的に見られる。このことから、強い正の相関を持つ特徴量の組み合わせが多く存在することがわかる。

対象データについては、特徴量の名称が匿名化されているので、具体的なセンサーの測定対象や方法については不明ではある。そのため、センサー間の測定対象については、ここでは不明だ。一般に、センサーを同じ機器に対して複数設置して取り付け位置によるセンサー情報の違いなどを検証する場合もある。こうしたことから推論すると、同じような機器に発生したイベントのデータを取得することになり、相関が強く見られていることなどが考えられる。

本件は171と特徴量が比較的多いことから、相関の強い特徴量についてまとめて独立性の高い特徴量を新たに設定することで、モデル構築の際の計算コストを削減と、予測精度の向上を図る。



特徴量間で値の分布が極端に異なる場合に、計算機乗の丸めの影響により精度が影響を受ける場合がある。そのため、スケール変換により、分布の幅による影響を回避することがある。本レポートの対象データについては、上述のボックス図に見る通りもともと-1から1の範囲にスケール調整されていた。そのため、スケール返還後も下記のボックス図に見る通り、分布の様子は変わらず、今回のデータについてはスケール変換の効果は少ないと判断した。今回のモデル構築にあたってはスケール変換は行わない。

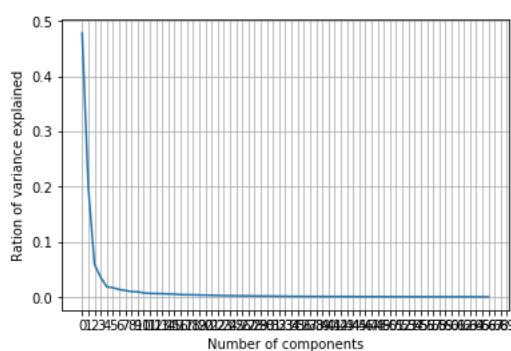
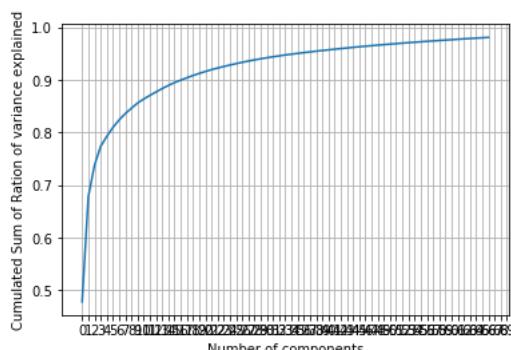


75パーセンタイルのデータは中央に集中しているものの、残りのデータについては幅広く広がっていることがわかる。PCA適用後のデータであり、クラスタ化されているという前提のもと、クラスタの中心から遠いデータをOutlierとして除外する。ここではOne Class SVMにより、Outlierの判定を行った。[https://scikit-learn.org/stable/modules/outlier\\_detection.html#outlier-detection](https://scikit-learn.org/stable/modules/outlier_detection.html#outlier-detection)

PCA変換するにあたり、98%の寄与率で特徴量を選んだ。この結果171の特徴量を67まで削減している。削減後の特徴量は互いに無相関となり、相互通関係数はほぼ0となっていることが見て取れる。

PCA変換後の特徴量に対し、ボックス図を描くと0付近に多くのデータが集約し、75パーセンタイルに含まれないデータの裾野が長いことが見て取れる。所与のトレーニングデータは均一なデータが多く、そこから外れるデータに多様性があることがわかる。

(60000, 67)

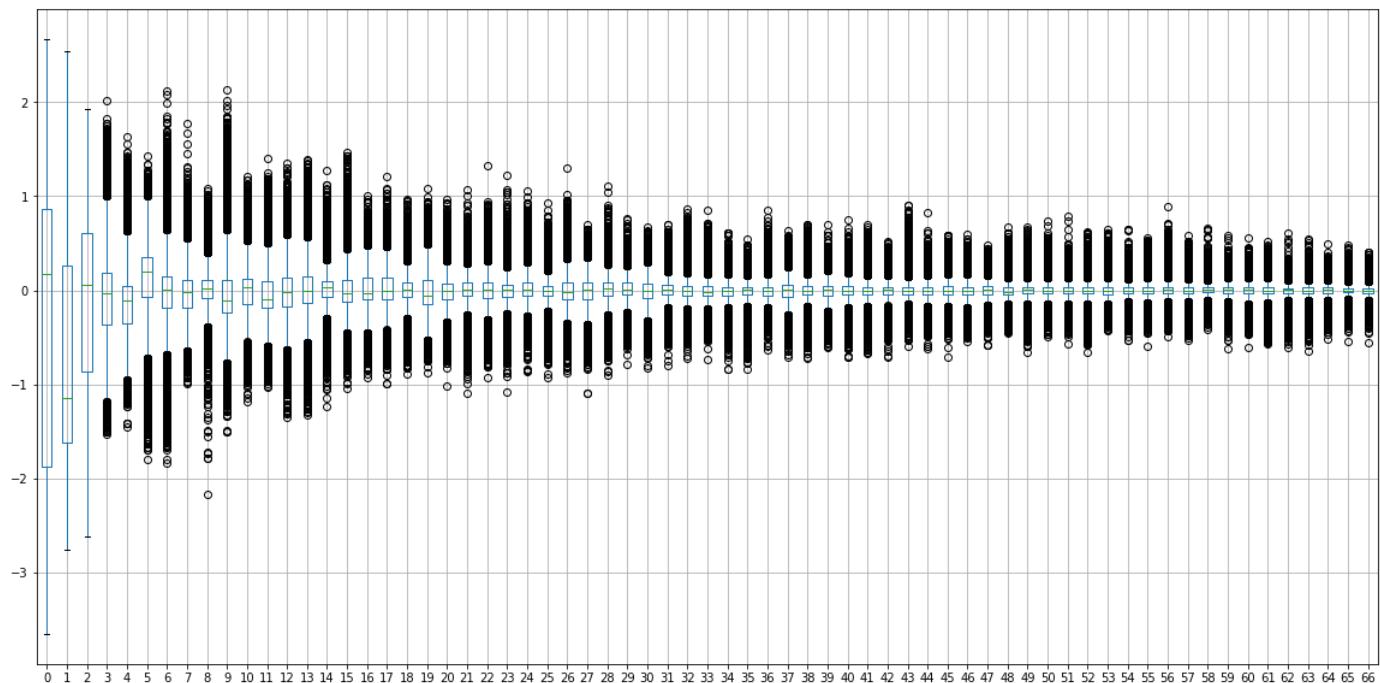
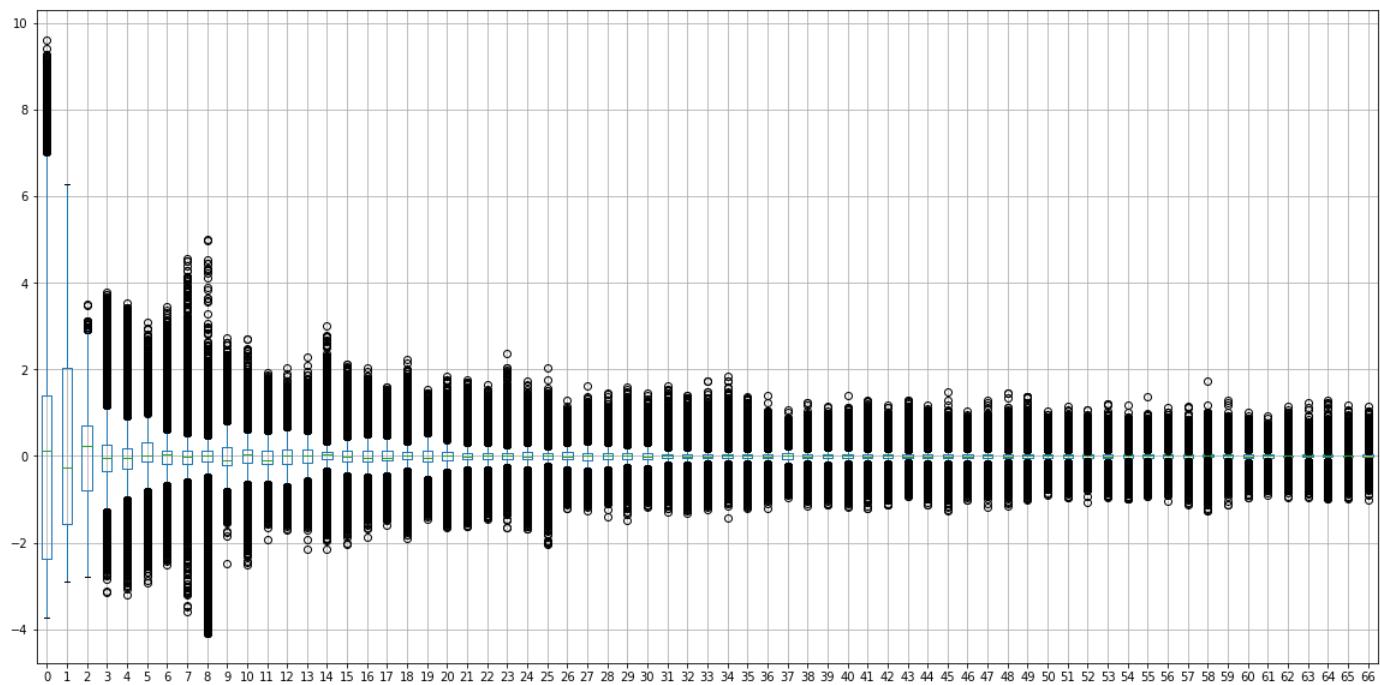


PCA(Primary Component Classifier)により171個の特徴量を削減し、互いに相関の低い67個の特徴量に変換することができたが、モデル構築時の計算コストや、精度の向上が期待できるものの、分類が適切に行えているか検証する。ポジティブクラス( consists of component failures for a specific component of the APS system)を分類するモデルの精度を高めることが本プロジェクトの目的である。これらのデータがPCAによる変換後にどのように分類されているか確認した。

One Class SVMによって50%を区切りにPositive ClassとNegative Classの割合をそれぞれ求めた60000件のデータを2つに分けると、One Class SVMによって除外されるOut in分類されたデータにはほとんどのPositiveクラスが分類されている。One Class SVMによる分類では、検出対象のPositiveクラスがほとんど除外されてしまうことがわかる。

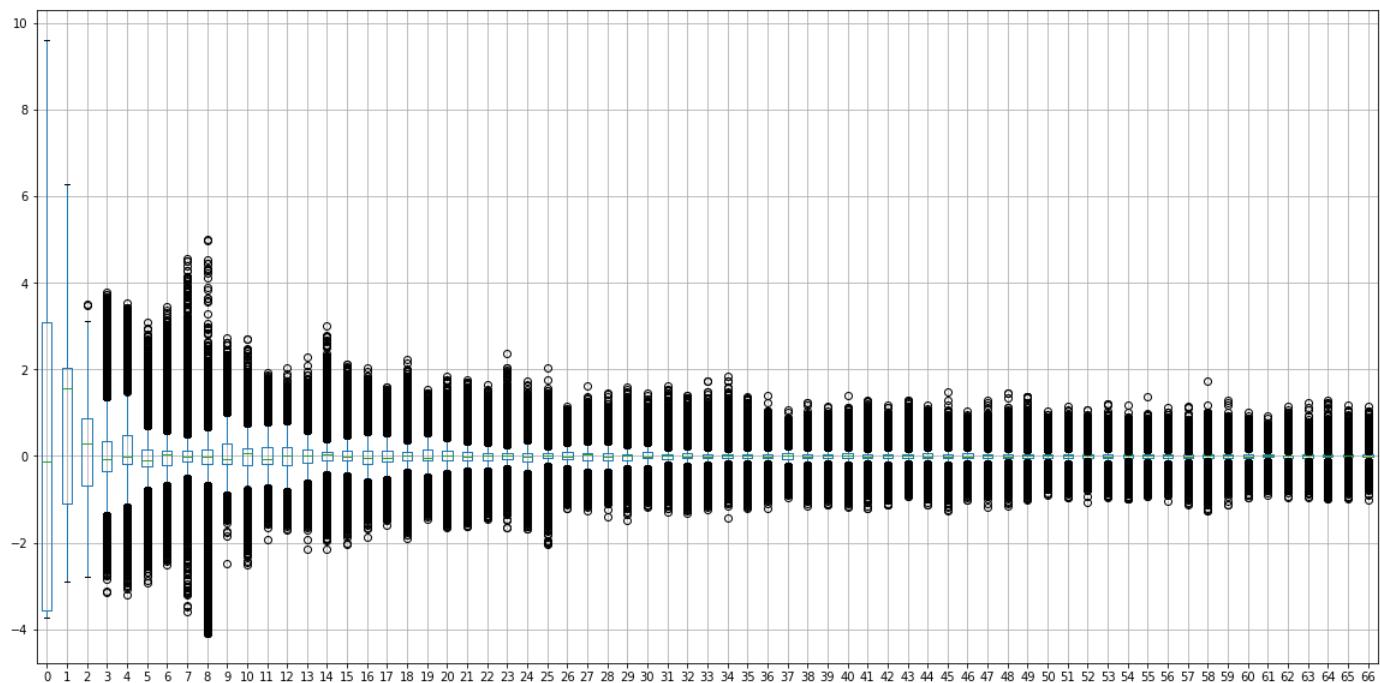
-	Positive	Negative
Out	991	29009
In	9	2991

センサーデータについては、ほとんどのデータは正常状態のデータである。今回のNegativeクラスのデータについては、何らかの車載コンポーネントに異常時に取得したデータではあるものの、対象となるAPSの異常が発生していない状況で取得されたデータだ。そのため、正常状態に偏ったデータが取得されていると考えられる。



```
0    29991  
1      9  
Name: class, dtype: int64
```

Positiveデータがほとんど除外されてしまったことがわかる



```
0    29009  
1     991  
Name: class, dtype: int64
```

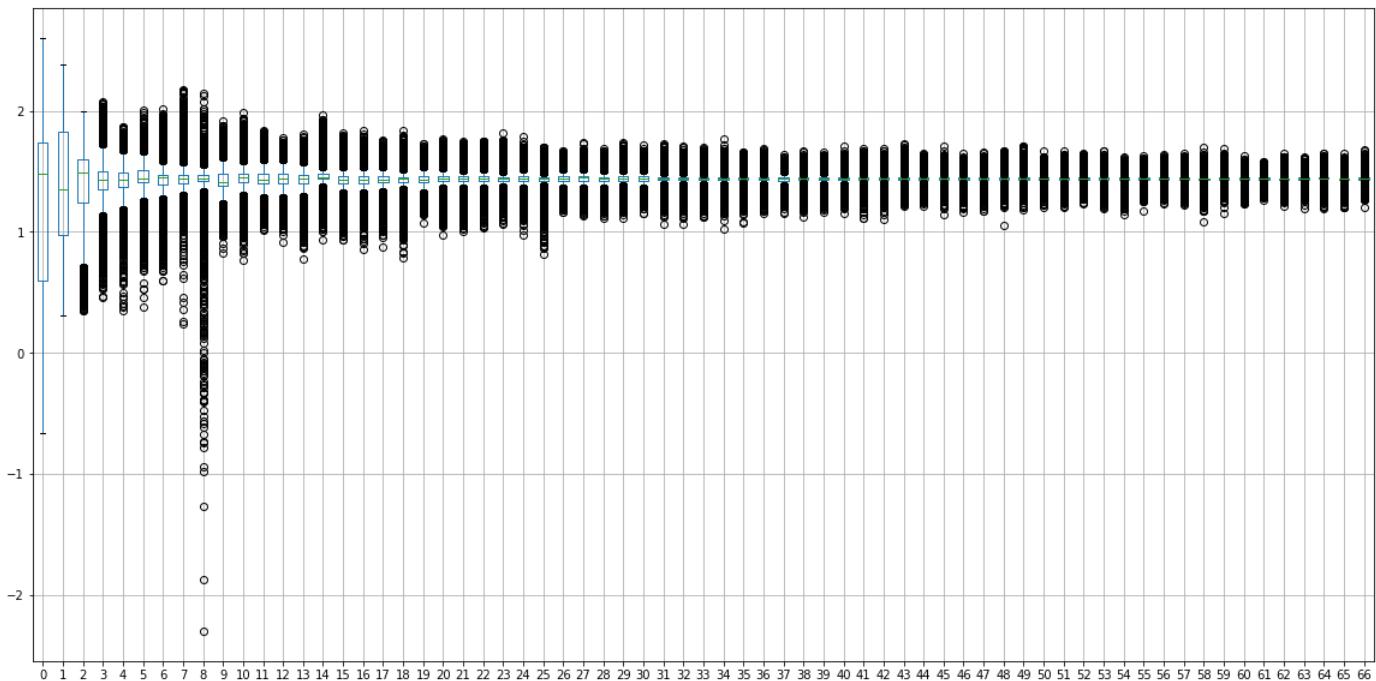
outlierは棄却せず、裾の広がりを緩和するために、対数変換してみる

```
X_reduced_test_data = pca.transform(X_test_given)  
  
MIN = min(np.min(X_reduced_data), np.min(X_reduced_test_data)) - 0.1  
MIN
```

```
-4.226051165194996
```

```
X_reduced_test_data_log = np.log(X_reduced_test_data - MIN)  
  
fig, ax = plt.subplots(1, 1, figsize=(20, 10))  
pd.DataFrame(X_reduced_test_data_log).boxplot(ax=ax)  
plt.show()
```

上記のことから、Outlierに重要なデータが含まれていると考え、Outlierの棄却は行わない。



## Evaluation Metrics

In evaluating the prediction logic, it is a challenge to extract failure events as much as possible while maintaining the operation rate. For that reason, it is necessary to accurately derive the judgment of correctness. Here, to make the balance of Confusion Matrix, adopt a model with a high result of scoring by ROC-AUC.

Accuracy, Precision, Recallのそれぞれを評価する。

## Benchmark Model

As benchmark, adopts 60% correct answer rate. This is based on a hearing from an interview myself conducted that the failure rate prevented by regular maintenance is about 60% by experience. It is worth considering the devided model in this project, if failure prediction is more than 60% Accuracy.

ただ、モデルの精度は極力高められたいことが望ましい。採用するべきデータ前処理ならびにXGBoostモデルのハイパーパラメータのベンチマークとして、RandomForestモデルと比較して評価する。

Random Forest uses a set of decision trees, and each tree represents some decision path to 'income' class, from features. Subset of features like 'capital-gain', 'education-num', 'age' and etc are used in each tree. The features picked up are different among trees. These features are used in the list of questions as a branch in the tree to reach from the ground to leaves(=income).

Usually, a single tree is not strong enough to be used in practice. To overcome this, Random Forest uses a lot of decision trees which are slightly different with each other. When we get a new answer from those trees, we take the majority vote of among the trees to get a final result. Compared to employing a single tree, you can reduce the proportion of incorrect results. By default, a Random Forest will use the square root of the number of features as the maximum features that it will look on any given branch. In our case we have total 103 features, so each decision will be the best of the 10(approximate) randomly selected features available.

```

Best ROC-AUC: 0.9890
accuracy score : 0.9896875
R-squared, coefficient of determination : 0.549
      precision    recall   f1-score   support
          0       0.99     1.00     0.99    15625
          1       0.85     0.68     0.76     375

   micro avg       0.99     0.99     0.99   16000
   macro avg       0.92     0.84     0.88   16000
weighted avg       0.99     0.99     0.99   16000

[[15579    46]
 [ 119   256]]


/opt/anaconda3/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:758: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
"of iterations.", ConvergenceWarning)

```

## Random Forest Classifier

```
Best ROC-AUC: 0.9885
accuracy score : 0.9858125
R-squared, coefficient of determination : 0.380
      precision    recall   f1-score   support
          0       0.99     1.00     0.99    15625
          1       0.97     0.41     0.57     375

   micro avg       0.99     0.99     0.99    16000
   macro avg       0.98     0.70     0.78    16000
weighted avg       0.99     0.99     0.98    16000

[[15621      4]
 [ 223    152]]
```

### XGBClassifier

```
Best ROC-AUC: 0.9944
accuracy score : 0.98975
R-squared, coefficient of determination : 0.552
      precision    recall   f1-score   support
          0       0.99     1.00     0.99    15625
          1       0.90     0.63     0.74     375

   micro avg       0.99     0.99     0.99    16000
   macro avg       0.95     0.81     0.87    16000
weighted avg       0.99     0.99     0.99    16000

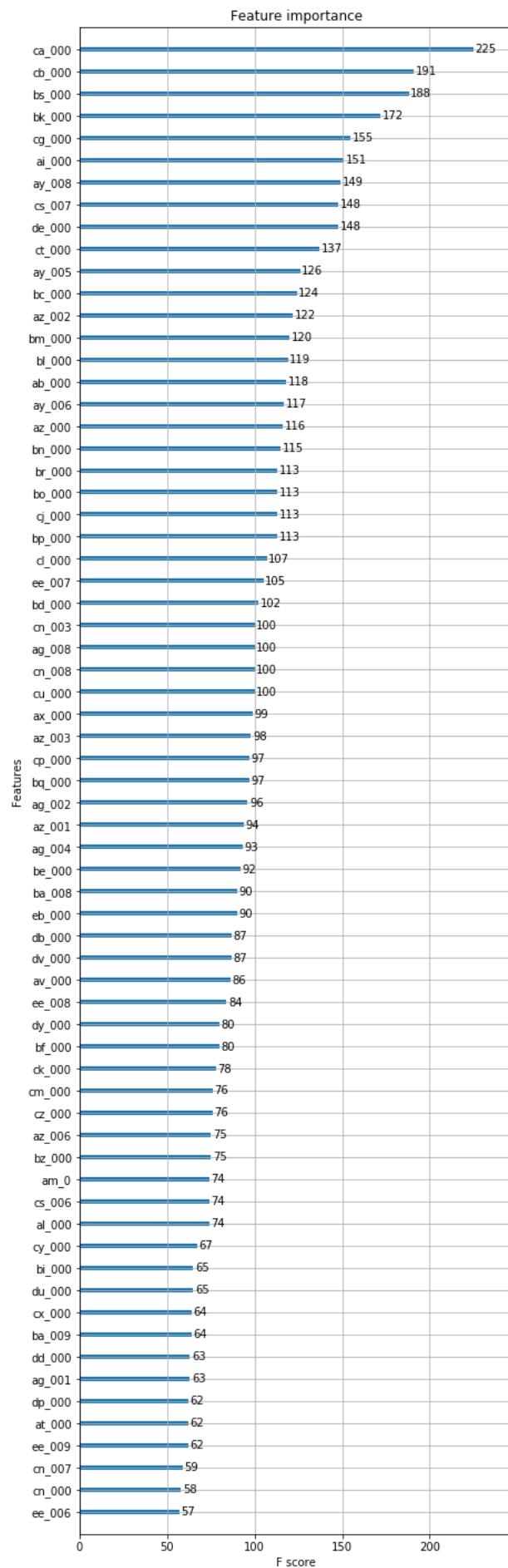
array([[15600,    25],
       [ 139,   236]])
```

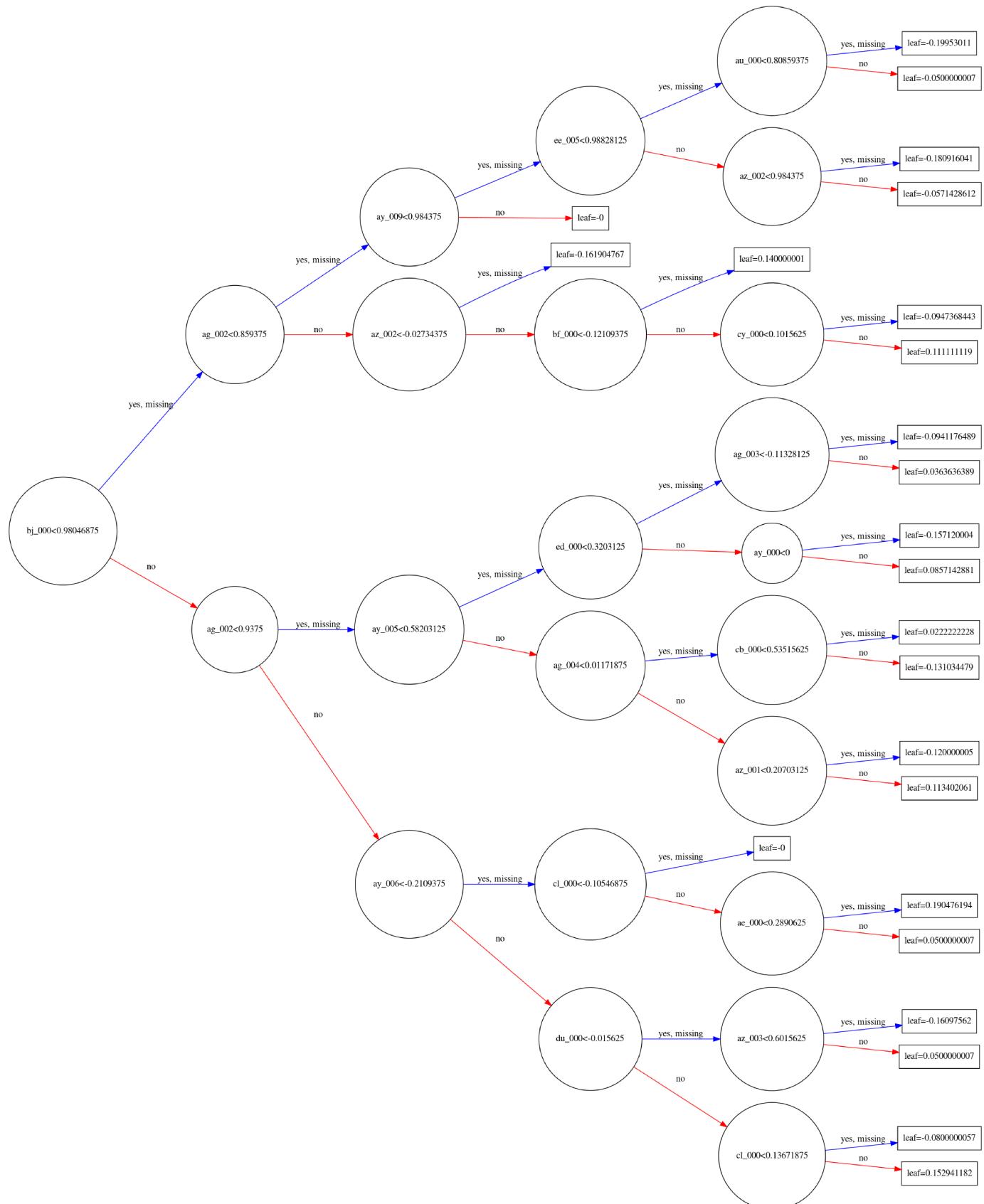
### XGBClassifier

```
Best ROC-AUC: 1.0000
accuracy score : 0.9925625
R-squared, coefficient of determination : 0.675
      precision    recall   f1-score   support
          0       0.99     1.00     1.00    15625
          1       0.92     0.75     0.83     375

   micro avg       0.99     0.99     0.99    16000
   macro avg       0.96     0.87     0.91    16000
weighted avg       0.99     0.99     0.99    16000

array([[15600,    25],
       [ 94,   281]])
```





PCAにより次元削減したデータセットに対するXGBoost

Best ROC-AUC: 0.9919 accuracy score : 0.9860625 R-squared, coefficient of determination : 0.391					
	precision	recall	f1-score	support	
0	0.99	1.00	0.99	15625	
1	0.87	0.48	0.62	375	
micro avg	0.99	0.99	0.99	16000	

```
macro avg      0.93      0.74      0.81      16000
weighted avg   0.98      0.99      0.98      16000
```

```
array([[15597,    28],
       [ 195, 180]])
```

```
%%time
predictor = xgb.XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                               colsample_bytree=0.8, gamma=0, learning_rate=0.1, max_delta_step=5,
                               max_depth=5, min_child_weight=1, missing=None, n_estimators=500,
                               n_jobs=1, nthread=4, objective='binary:logistic', random_state=0,
                               reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=42, silent=True,
                               subsample=0.5, tree_method='exact', verbose=10)
```

```
predictor.fit(X_train, y_train)
```

```
CPU times: user 5min 42s, sys: 547 ms, total: 5min 43s
Wall time: 1min 41s
```

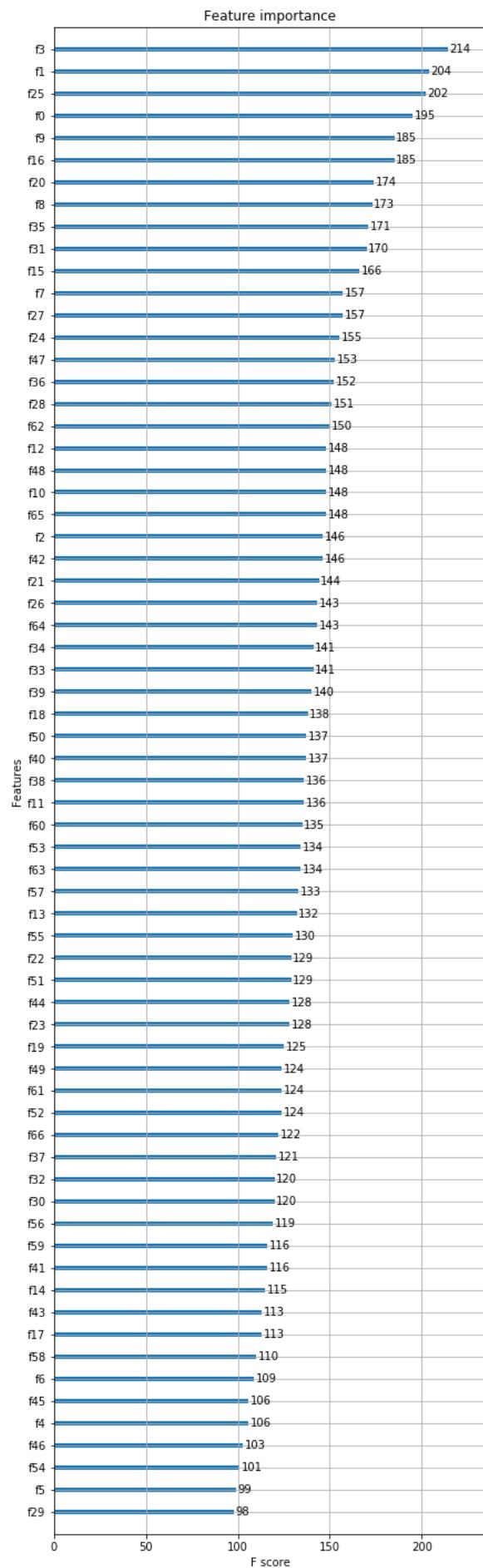
```
score = predictor.predict_proba(X_train)
print('Best ROC-AUC: {:.4f}'.format(roc_auc_score(y_train, score[:, 1], average='macro')))
predict = predictor.predict(X_test)
print("accuracy score : {}".format(accuracy_score(y_test_given, predict)))
print("R-squared, coefficient of determination : {:.3f}".format(r2_score(y_test, predict)))
print(classification_report(y_true = y_test, y_pred = predict ))
confusion_matrix(y_true = y_test, y_pred = predict )
```

```
Best ROC-AUC: 1.0000
accuracy score : 0.9889375
R-squared, coefficient of determination : 0.517
      precision    recall  f1-score   support
          0         0.99     1.00      0.99    15625
          1         0.89     0.60      0.72      375

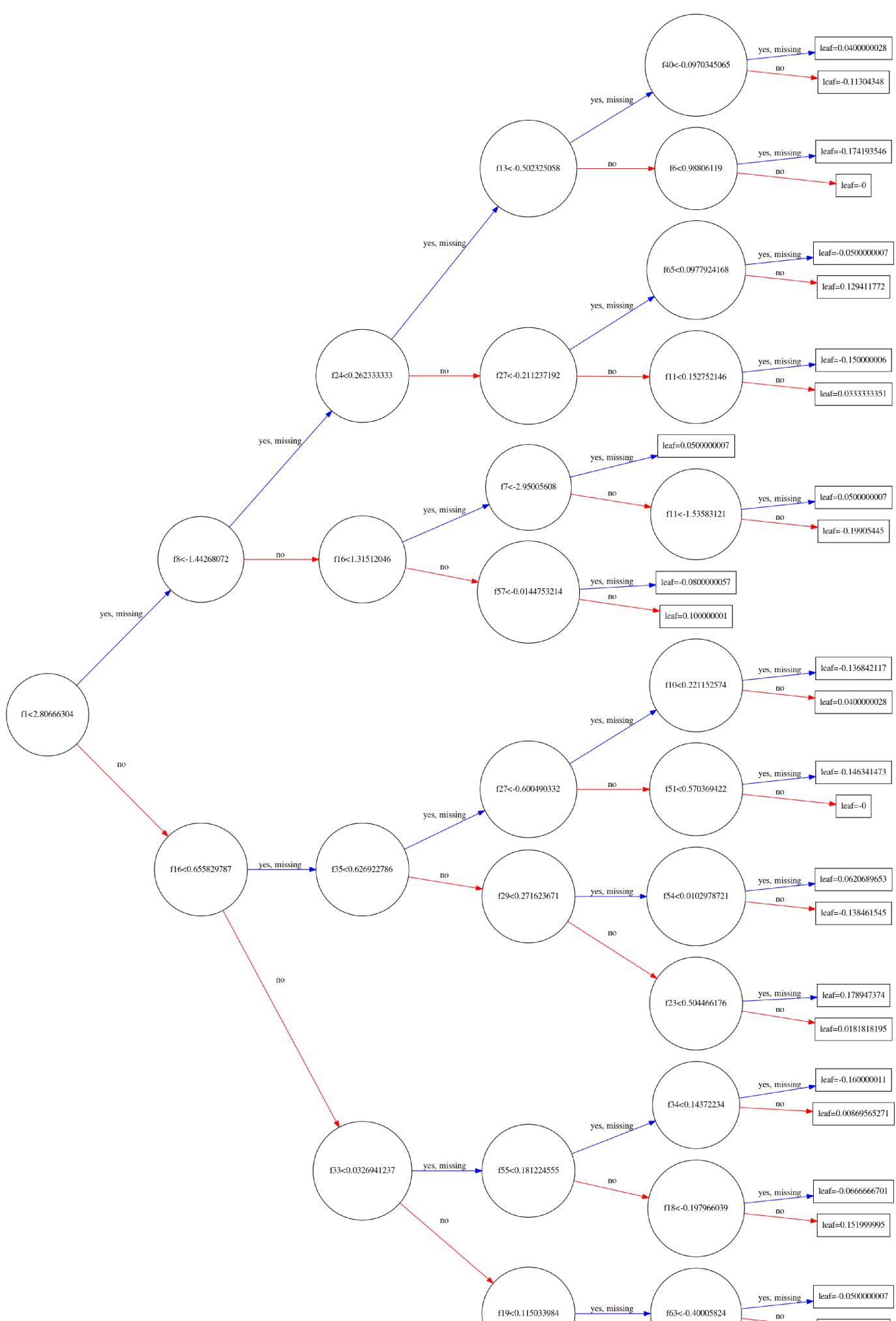
   micro avg      0.99      0.99      0.99      16000
   macro avg      0.94      0.80      0.86      16000
weighted avg     0.99      0.99      0.99      16000
```

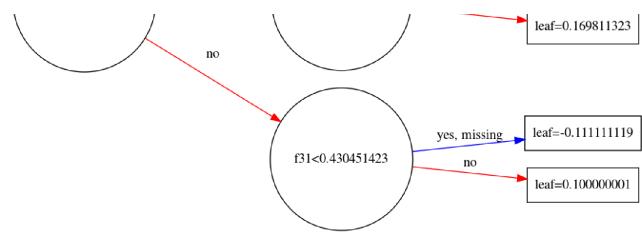
```
array([[15598,    27],
       [ 150, 225]])
```

```
fig, ax = plt.subplots(1, 1, figsize=(7, 25))
plot_importance(predictor, max_num_features = pca.n_components_, ax=ax)
plt.show()
```



```
plot_tree(predictor, rankdir='LR')
fig = plt.gcf()
fig.set_size_inches(150, 100)
plt.show()
```

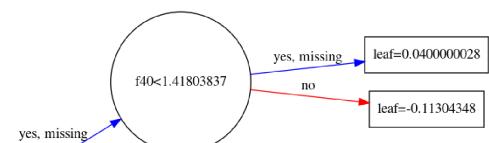
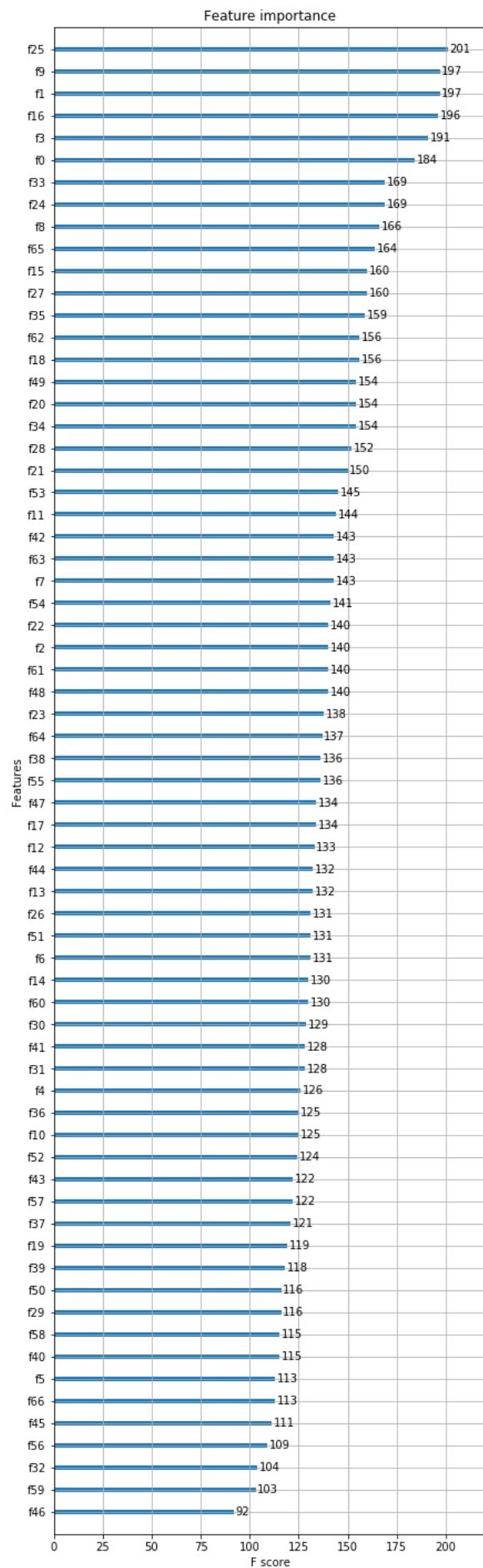


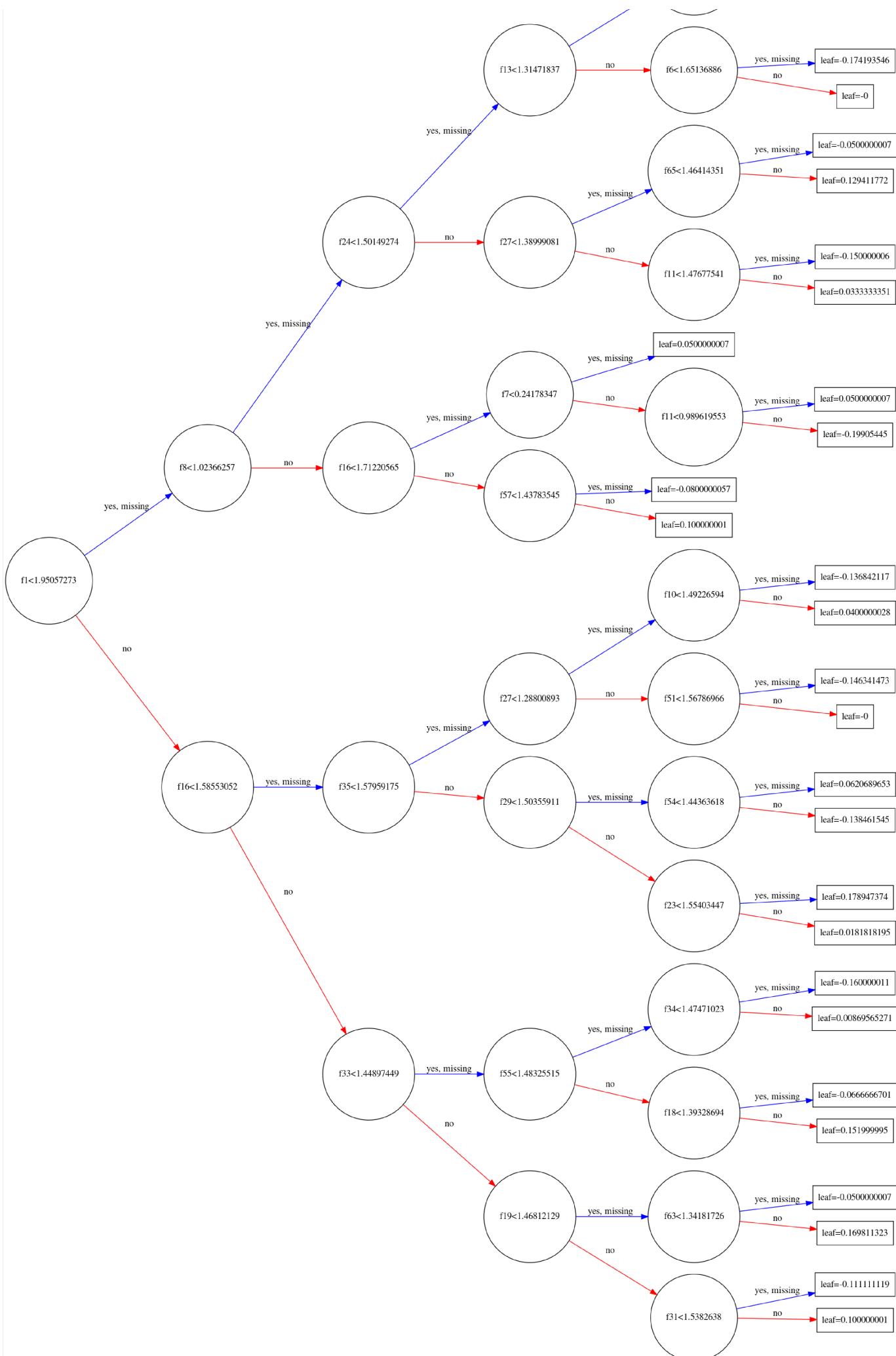


### Log TransformによりSkewならびに裾野のデータを調整したXGBoost

```
Best ROC-AUC: 1.0000
accuracy score : 0.9884375
R-squared, coefficient of determination : 0.495
precision      recall    f1-score   support
0            0.99     1.00      0.99    15625
1            0.89     0.58      0.70     375
micro avg       0.99     0.99      0.99    16000
macro avg       0.94     0.79      0.85    16000
weighted avg    0.99     0.99      0.99    16000
```

```
array([[15598,      27],
       [ 158,    217]])
```





## 評価

### Implementation: Define a Performance Metric

It is difficult to measure the quality of a given model without quantifying its performance over training and testing. This is typically done using some type of performance metric, whether it is through calculating some type of error, the goodness of fit, or some other useful measurement. For this project, you will be calculating the coefficient of determination, R2, to quantify your model's performance. The coefficient of determination for a model is a useful statistic in regression analysis, as it often describes how "good" that model is at making predictions.

The values for R2 range from 0 to 1, which captures the percentage of squared correlation between the predicted and actual values of the target variable. A model with an R2 of 0 is no better than a model that always predicts the mean of the target variable, whereas a model with an R2 of 1 perfectly predicts the target variable. Any value between 0 and 1 indicates what percentage of the target variable, using this model, can be explained by the features. A model can be given a negative R2 as well, which indicates that the model is arbitrarily worse than one that always predicts the mean of the target variable.

For the performance\_metric function in the code cell below, you will need to implement the following:

Use `r2_score` from `sklearn.metrics` to perform a performance calculation between `y_true` and `y_predict`. Assign the performance score to the `score` variable.

R2 score of 0 means that the dependent variable cannot be predicted from the independent variable.

R2 score of 1 means the dependent variable can be predicted from the independent variable.

R2 score between 0 and 1 indicates the extent to which the dependent variable is predictable. An

R2 score of 0.40 means that 40 percent of the variance in Y is predictable from X.

文献Tarald O. Kvalseth: "Cautionary Note about R2", The American Statistician Vol. 39, No. 4, Part 1 (Nov., 1985), pp. 279-285 の（1）の定義を採用する。

R-squared, coefficient of determination  $\frac{\sum_{i=0}^{n-1} (y_i - \hat{y})^2}{\sum_{i=0}^{n-1} (y_i - \bar{y})^2}$

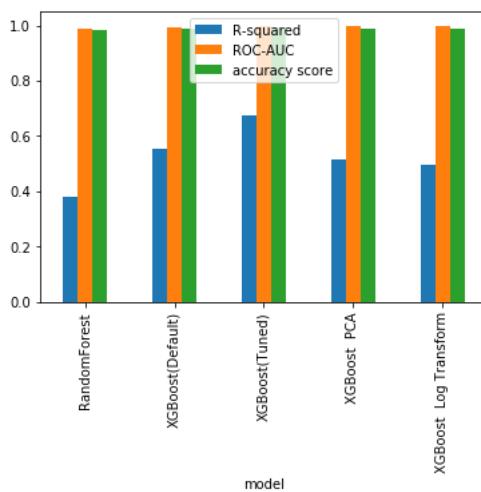
$y$  : Actual Value  $\hat{y}$  : Predicted Value  $\bar{y}$  : Average Value  $n$  : the number of samples

ROC-AUC and accuracy score are close to 1.0 for any models benchmarked. But R-squared, coefficient of determination is best 0.675 only for XGBoost.

XGBoost improved the predictio model against Rando Forest in this case. But R2 score of 0.40 means that 40 percent of the variance in Y is predictable from X. Which is lower than the regular maintenance measure.

RainForestよりも良い結果が得られているが、データの前処理を行わないほうが、R2スコアが良い傾向にあることがわかる。また、ハイパーパラメータを調整することでさらにR2スコアが改善されることが予測される。今回の対象データは、検出したい対象クラスが分布の中心の75パーセンタイルから外れる少数派に含まれるという、データ分布上の特徴があった。完全にランダムなデータではなく、測定対象固有の確率分布に従っている可能性がある。そのため、正規分布を想定したデータの前処理はモデルの精度を高めることにはつながらず、むしろ分布の少数派のデータは積極的に活用したほうがよいことが推測される。

これらのことから、当初の60%以上の正確さを期して、データ前処理を行わずに、XGBoostで最適なハイパーパラメータを調整することにする。



Precision,Recallなどみてもデータ前処理を行わないXGBoostが高い性能を発揮していることがわかる。

### RandomForest

	precision	recall	f1-score	support
0	0.99	1.00	0.99	15625
1	0.97	0.41	0.57	375

### XGBoost

	precision	recall	f1-score	support
0	0.99	1.00	1.00	15625
1	0.92	0.75	0.83	375

**XGBoost PCA**

	precision	recall	f1-score	support
0	0.99	1.00	0.99	15625
1	0.89	0.60	0.72	375

**XGBoost Log Transform**

	precision	recall	f1-score	support
0	0.99	1.00	0.99	15625
1	0.89	0.58	0.70	375

**Seek hyperparameters**

[https://xgboost.readthedocs.io/en/latest/python/python\\_api.html#xgboost.XGBClassifier](https://xgboost.readthedocs.io/en/latest/python/python_api.html#xgboost.XGBClassifier)

訓練データとテストデータについては、上記を踏まえて前処理を施さない、所与のデータを活用する。

ハイパーパラメータの決定にあたり、本プロジェクトは以下の探索を行った。事前に、いくつかのパラメータを調整し、改善効果の大きかった`max_depth (int)`, `n_estimators (int)`\*に対象を絞っている。

Parameter	Description	Search Range
<code>max_depth (int)</code>	Maximum tree depth for base learners.	[5,6,7]
<code>learning_rate (float)</code>	Boosting learning rate (xgb's "eta")	0.1(Fixed)
<code>n_estimators (int)</code>	Number of boosted trees to fit.	[300, 500, 700]
<code>objective (string or callable)</code>	Specify the learning task and the corresponding learning objective or a custom objective function to be used.	['binary:logistic']
<code>booster (string)</code>	Specify which booster to use	gbtree
<code>gamma (float)</code>	Minimum loss reduction required to make a further partition on a leaf node of the tree.	[0.0] (Fixed)
<code>min_child_weight (int)</code>	Minimum sum of instance weight(hessian) needed in a child.	[1]
<code>max_delta_step (int)</code>	Maximum delta step we allow each tree's weight estimation to be.	[5]
<code>subsample (float)</code>	Subsample ratio of the training instance.	[0.5] (Fixed)
<code>colsample_bytree (float)</code>	Subsample ratio of columns when constructing each tree.	[0.8]

```

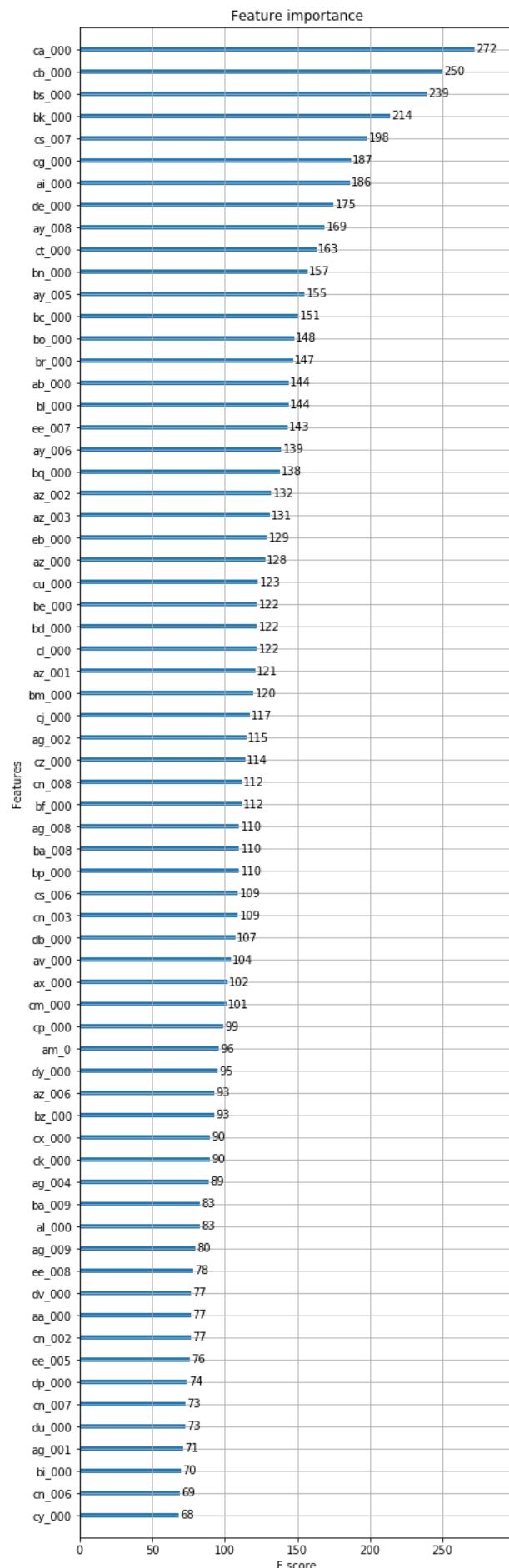
Best ROC-AUC: 1.0000
accuracy score : 0.99275
R-squared, coefficient of determination : 0.683
      precision    recall   f1-score   support
0       0.99     1.00     1.00    15625
1       0.94     0.74     0.83     375

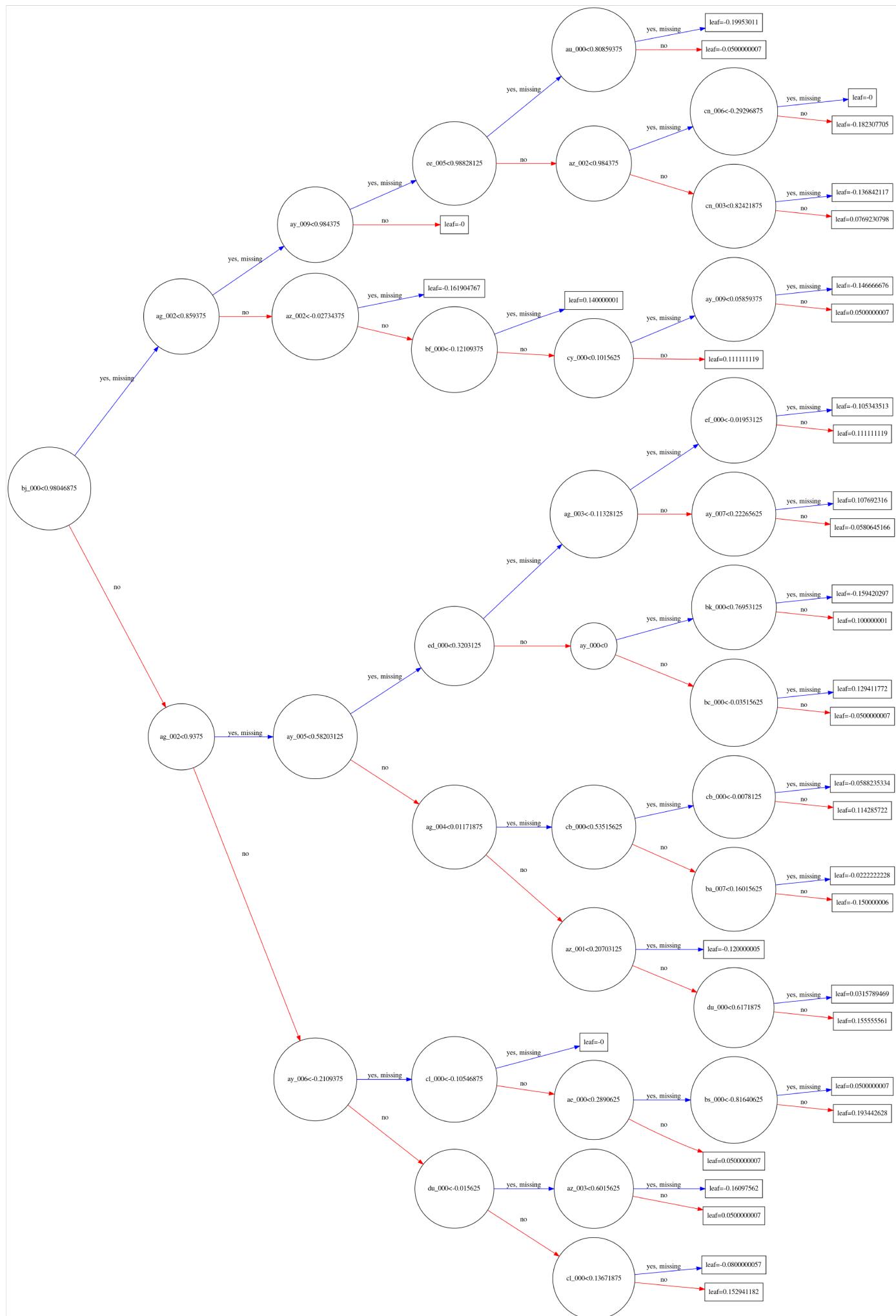
   micro avg     0.99     0.99     0.99    16000
   macro avg     0.97     0.87     0.91    16000
 weighted avg    0.99     0.99     0.99    16000

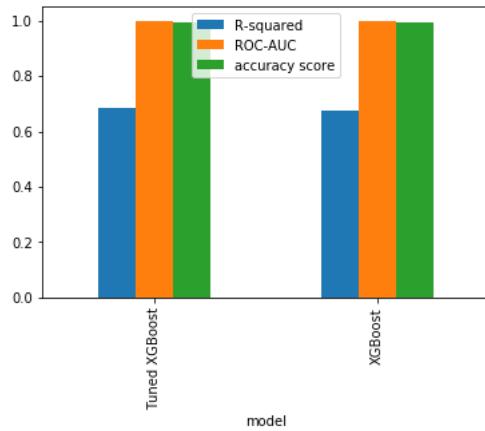
array([[15607,      18,
       98,  277]])

Best parameters: {'colsample_bytree': 0.8, 'gamma': 0.0, 'learning_rate': 0.1, 'max_delta_step': 5, 'max_depth': 6,
'min_child_weight': 1, 'n_estimators': 500, 'nthread': 4, 'objective': 'binary:logistic', 'scale_pos_weight': 1,
'seed': 42, 'subsample': 0.5, 'tree_method': 'exact', 'verbose': 10}
Best auroc score: 0.716

```

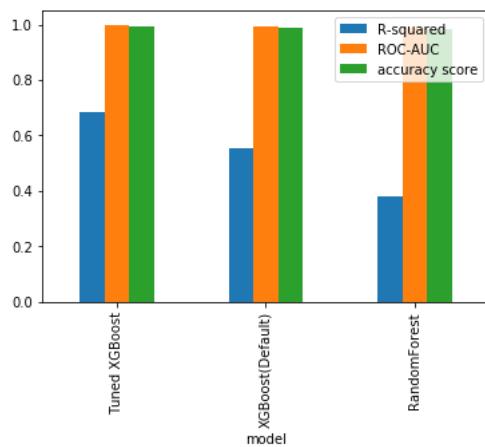






## V. Conclusion

XGBoostを採用することで、ベンチマークに用いたRandom Forestよりも高い適合度の分類予測モデルを構築することができた。ハイパーパラメータをチューニングすることで、モデルの適合度を上げることができた。



## Reflection

今回は、APSまわりのセンサーから取得したデータを元に、以上発生の有無を予測するモデルを作成した。センサーデータの特徴として、大多数のデータは平常時のデータであり、多くの前処理で外れ値に分類されるような分布の中心から離れたデータも予測モデルの構築に重要な役割をはたすことが見てとれた。また、そのため、特微量間で強い相関が見られるものの、PCA等によりクラスタ化して、特徴を表す軸を見出し、特微量を減らすといった処理をするよりも、そのままの特徴データを活用するほうがよい予測モデルを構築することができた。

このことは、クラス分けに寄与する決定木が導かれたら、PCAで必要となる特微量の変換と逆変換といったステップを経ずに、重要な特微量、すなわちセンサーを導けるといった点で利便性が高い。トラックなどは、センサーが高価であり、設置に多くの労力とコスト、時間を要することから、XGBoostを活用した今回のアプローチは、車載センサーによる故障予測の分野で汎用的に役立つことが期待される。

このプロジェクトでは最後に、XGBoostのハイパーパラメータの調整を行った。調整を行うことでモデルの適合度を高めることができた。ただし、全パラメータにわたって、調整を行うことは計算資源の制約から今回は見送った。

## Improvement

今回のデータ・セットはあらじめ検出対象のクラスの割合が調整されていた可能性がある。通常のセンサーデータは、平常時のデータが圧倒的に多く生成される。そのため異常時のデータがより少ない、調整前のデータに対しても今回のアプローチが有効であるか追加で検証することが、汎用的に実用化するには必要だと考えられる。また、XGBoostはハイパーパラメータの調整に時間がかかるアプローチであり、そのためにGPUの活用など、計算機資源の強化による速度向上が期待できる。