Maintanence_Prediction

September 28, 2023

1 Maintenance-Prediction Notebook

- By Malay Vyas

[96]:
```
import pickle
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import tree
import pandas_profiling as pp
from tpot import TPOTClassifier
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import log_loss, f1_score
from ydata_profiling import ProfileReport as pr
from lazypredict.Supervised import LazyClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import ExtraTreesClassifier
```

[73]:
```
df1 = pd.read_csv('original_dataset.csv')
```

[74]:
```
df2 = pd.read_csv('generated_dataset.csv')
```

[75]:
```
df1 = df1.drop(['UDI', 'Product ID', 'TWF','HDF','PWF','OSF','RNF'], axis=1)
```

[76]:
```
df2 = df2.drop(['UDI', 'Product ID', 'TWF','HDF','PWF','OSF','RNF'], axis=1)
```

[77]:
```
df1.describe()
```

[77]:
```
       Air temperature [K]  Process temperature [K]  Rotational speed [rpm]  \
count             10000.00                 10000.00                10000.00
mean                300.00                   310.01                 1538.78
std                   2.00                     1.48                  179.28
min                 295.30                   305.70                 1168.00
25%                 298.30                   308.80                 1423.00
50%                 300.10                   310.10                 1503.00
75%                 301.50                   311.10                 1612.00
```

```
max                  304.50                   313.80                2886.00

       Torque [Nm]  Tool wear [min]  Machine failure
count     10000.00         10000.00         10000.00
mean         39.99           107.95             0.03
std           9.97            63.65             0.18
min           3.80             0.00             0.00
25%          33.20            53.00             0.00
50%          40.10           108.00             0.00
75%          46.80           162.00             0.00
max          76.60           253.00             1.00
```

[78]:

```
df2.describe()
```

[78]:

```
       Air temperature [K]  Process temperature [K]  Rotational speed [rpm]  \
count              5000.00                  5000.00                 5000.00
mean                300.14                   309.89                 1569.97
std                   1.97                     1.53                  229.71
min                 295.30                   305.70                 1217.00
25%                 298.50                   308.80                 1416.00
50%                 300.30                   309.80                 1505.00
75%                 301.60                   311.10                 1668.00
max                 304.50                   313.70                 2886.00

       Torque [Nm]  Tool wear [min]  Machine failure
count      5000.00          5000.00          5000.00
mean         40.51           101.90             0.02
std          12.45            59.03             0.14
min           6.40             0.00             0.00
25%          31.40            55.00             0.00
50%          41.10            85.00             0.00
75%          49.00           152.00             0.00
max          76.60           243.00             1.00
```

[56]:

```
pr(df1, title="Original Datset Report")
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

<IPython.core.display.HTML object>
```

[56]:

[57]:

```
pr(df2, title="Original Datset Report")
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]
```

```
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]
```

```
Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
```

```
<IPython.core.display.HTML object>
```

[57]:

[79]:
```
le = LabelEncoder()
df1["Type"] = le.fit_transform(df1["Type"])
```

[80]:
```
y = df1["Machine failure"]
x = df1.drop(["Machine failure"],axis=1)
```

[81]:
```
x_train,x_test,y_train,y_test =␣
 ↪train_test_split(x,y,random_state=42,test_size=0.2)
```

[82]:
```
clf = LazyClassifier(verbose=0, ignore_warnings=True, custom_metric=None)
models,predictions = clf.fit(x_train, x_test, y_train, y_test)
```

```
100%|     | 29/29 [00:11<00:00,  2.50it/s]
```

```
[LightGBM] [Info] Number of positive: 278, number of negative: 7722
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.000269 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 928
[LightGBM] [Info] Number of data points in the train set: 8000, number of used
features: 6
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.034750 -> initscore=-3.324208
[LightGBM] [Info] Start training from score -3.324208
```

[83]:
```
models
```

[83]:
```
                               Accuracy  Balanced Accuracy  ROC AUC  F1 Score  \
Model
DecisionTreeClassifier             0.98               0.84     0.84      0.98
XGBClassifier                      0.98               0.83     0.83      0.98
LGBMClassifier                     0.99               0.83     0.83      0.99
BaggingClassifier                  0.99               0.83     0.83      0.99
RandomForestClassifier             0.98               0.79     0.79      0.98
NearestCentroid                    0.72               0.77     0.77      0.81
AdaBoostClassifier                 0.98               0.74     0.74      0.98
LabelSpreading                     0.98               0.73     0.73      0.97
LabelPropagation                   0.97               0.73     0.73      0.97
ExtraTreeClassifier                0.97               0.73     0.73      0.97
```

```
PassiveAggressiveClassifier        0.96               0.72     0.72      0.96
ExtraTreesClassifier               0.98               0.71     0.71      0.98
Perceptron                         0.91               0.70     0.70      0.93
LinearDiscriminantAnalysis         0.97               0.69     0.69      0.97
KNeighborsClassifier               0.98               0.69     0.69      0.98
QuadraticDiscriminantAnalysis      0.96               0.66     0.66      0.96
SVC                                0.98               0.65     0.65      0.97
LogisticRegression                 0.97               0.63     0.63      0.97
CalibratedClassifierCV             0.97               0.62     0.62      0.97
GaussianNB                         0.96               0.60     0.60      0.96
LinearSVC                          0.97               0.56     0.56      0.96
SGDClassifier                      0.97               0.52     0.52      0.96
DummyClassifier                    0.97               0.50     0.50      0.95
RidgeClassifier                    0.97               0.50     0.50      0.95
RidgeClassifierCV                  0.97               0.50     0.50      0.95
BernoulliNB                        0.97               0.50     0.50      0.95

                               Time Taken
Model
DecisionTreeClassifier               0.06
XGBClassifier                        0.52
LGBMClassifier                       0.16
BaggingClassifier                    0.22
RandomForestClassifier               0.94
NearestCentroid                      0.02
AdaBoostClassifier                   0.34
LabelSpreading                       4.85
LabelPropagation                     2.15
ExtraTreeClassifier                  0.02
PassiveAggressiveClassifier          0.03
ExtraTreesClassifier                 0.43
Perceptron                           0.03
LinearDiscriminantAnalysis           0.07
KNeighborsClassifier                 0.17
QuadraticDiscriminantAnalysis        0.03
SVC                                  0.32
LogisticRegression                   0.04
CalibratedClassifierCV               0.73
GaussianNB                           0.02
LinearSVC                            0.20
SGDClassifier                        0.05
DummyClassifier                      0.02
RidgeClassifier                      0.02
RidgeClassifierCV                    0.03
BernoulliNB                          0.02
```

[84]:

```
df_sum = models.drop(["Time Taken"], axis=1).sum(axis=1)
```

[85]:

```
df = models.iloc[:,:-1].sum(axis=1)
df_sum1 = df/4
```

[86]:

```
df_sum1.sort_values(ascending=False)
```

[86]:

```
Model
XGBClassifier                    0.91
DecisionTreeClassifier           0.91
LGBMClassifier                   0.91
BaggingClassifier                0.91
RandomForestClassifier           0.89
AdaBoostClassifier               0.86
LabelSpreading                   0.85
LabelPropagation                 0.85
ExtraTreeClassifier              0.85
ExtraTreesClassifier             0.85
PassiveAggressiveClassifier      0.84
KNeighborsClassifier             0.83
LinearDiscriminantAnalysis       0.83
Perceptron                       0.81
SVC                              0.81
QuadraticDiscriminantAnalysis    0.81
LogisticRegression               0.80
CalibratedClassifierCV           0.80
GaussianNB                       0.78
NearestCentroid                  0.77
LinearSVC                        0.76
SGDClassifier                    0.74
DummyClassifier                  0.73
RidgeClassifier                  0.73
RidgeClassifierCV                0.73
BernoulliNB                      0.73
dtype: float64
```

[88]:

```
pipeline_optimizer = TPOTClassifier(generations=5, population_size=20,␣
 ↪cv=5,random_state=42, verbosity=2)
pipeline_optimizer.fit(x_train, y_train)
print(pipeline_optimizer.score(x_test, y_test))
pipeline_optimizer.export('tpot_exported_pipeline.py')
```

```
Optimization Progress:   0%|          | 0/120 [00:00<?, ?pipeline/s]


Generation 1 - Current best internal CV score: 0.9838749999999999

Generation 2 - Current best internal CV score: 0.9838749999999999

Generation 3 - Current best internal CV score: 0.983875
Generation 4 - Current best internal CV score: 0.9848750000000001

Generation 5 - Current best internal CV score: 0.9848750000000001

Best pipeline: ExtraTreesClassifier(input_matrix, bootstrap=False,
criterion=entropy, max_features=0.8500000000000001, min_samples_leaf=2,
min_samples_split=6, n_estimators=100)
0.9865
```

```
[89]: etc = ExtraTreesClassifier(bootstrap=False, criterion='entropy', max_features=0.
        ↪8500000000000001, min_samples_leaf=2, min_samples_split=6, n_estimators=100)
```

```
[90]: etc.fit(x_train,y_train)
```

```
[90]: ExtraTreesClassifier(criterion='entropy', max_features=0.8500000000000001,
                           min_samples_leaf=2, min_samples_split=6)
```

```
[91]: y_pred = etc.predict(x_test)
```

```
[92]: ll1 = cross_val_score(clf, x_test,y_test, cv=5)
      ll1
```

```
[92]: array([0.9725, 0.965 , 0.9775, 0.965 , 0.9725])
```

```
[98]: Model = 'Model.sav'
      pickle.dump(etc, open(Model, 'wb'))
```