Classification

Data import

```
import pandas as pd
import os
import numpy as np
```

```
traindata_url = 'https://bitbucket.org/hyuk125/lg_dic/raw/889649d1bc273bf53967
testdata_url = 'https://bitbucket.org/hyuk125/lg_dic/raw/889649d1bc273bf53967c
train_data = pd.read_csv(traindata_url)
test_data = pd.read_csv(testdata_url)
```

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

le.fit(train_data.label == 5)

train_data.label = le.transform(train_data.label == 5)
test_data.label = le.transform(test_data.label == 5)
```

```
X_train = train_data.values[:, 1:]
y_train = train_data.values[:, 0]
X_test = test_data.values[:, 1:]
y_test = test_data.values[:, 0]
```

```
X = np.r_[X_train, X_test]
y = np.r_[y_train, y_test]
```

Classification 모델

Data 처리

Training / test 분할

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

Randomforest classification 모델 학습

```
from sklearn.ensemble import RandomForestClassifier
```

```
model = RandomForestClassifier(n_estimators=50, max_leaf_nodes=16, n_jobs=-1)
model.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=16, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=50, n_jobs=-1,
                       oob_score=False, random_state=None, verbose=0,
                       warm_start=False)
```

Model Test

정확도 판단

Confution matrix

```
from sklearn.metrics import confusion_matrix
```

```
predict = model.predict(X_test)
```

```
matrix = confusion_matrix(y_test, predict)
matrix
```

```
array([[15855,     7],
       [  942,   696]])
```

Precision, recall

```
from sklearn.metrics import precision_score, recall_score
```

```
print(precision_score(y_test, predict, average=None))
print('average: ', precision_score(y_test, predict, average='weighted'))
```

```
[0.94391856 0.99004267]
average:  0.9482357742708069
```

```
print(recall_score(y_test, predict, average=None))
print('average: ', recall_score(y_test, predict, average='macro'))
```

```
[0.99955869 0.42490842]
average:  0.7122335593209379
```