Ex 1: Glass.data

Cho dữ liệu glass.data.txt

Sử dụng thuật toán ADABoosting/XGBoost & thuật toán cơ sở để dự đoán loại kính dựa trên các thông tin được cung cấp

1. Đọc dữ liệu và gán cho biến data. Xem thông tin data: shape, type, head(), tail(), info. Tiền xử lý dữ liệu (nếu cần)
2. Tạo inputs data với các cột trừ cột type of class, và outputs data với 1 cột là type of class
3. Từ inputs data và outputs data => Tạo X_train, X_test, y_train, y_test với tỷ lệ 70-30
4. Thực hiện ADABoosting/XGBoost với X_train, y_train
5. Dự đoán y từ X_test => so sánh với y_test
6. Đánh giá mô hình => Nhận xét
7. Ghi mô hình (nếu mô hình tốt sau khi đánh giá)

Attribute Information:

1. Id number: 1 to 214
2. RI: refractive index
3. Na: Sodium (unit measurement: weight percent in corresponding oxide, as are attributes 4-10)
4. Mg: Magnesium
5. Al: Aluminum
6. Si: Silicon
7. K: Potassium
8. Ca: Calcium
9. Ba: Barium
10. Fe: Iron
11. Type of glass: (class attribute)

-- 1 building_windows_float_processed -- 2 building_windows_non_float_processed -- 3 vehicle_windows_float_processed -- 4 vehicle_windows_non_float_processed (none in this database) -- 5 containers -- 6 tableware -- 7 headlamps

In [1]:
```python
# from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
```

In [2]:
```python
# %cd '/content/gdrive/My Drive/LDS6_MachineLearning/practice_2023/Chapter8_Boosting/'
```

In [3]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

In [4]:
```python
# import some data to play with
data = pd.read_csv("glass.data.txt", sep=",", header=None)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 11 columns):
0     214 non-null int64
1     214 non-null float64
2     214 non-null float64
3     214 non-null float64
4     214 non-null float64
5     214 non-null float64
6     214 non-null float64
7     214 non-null float64
8     214 non-null float64
9     214 non-null float64
10    214 non-null int64
dtypes: float64(9), int64(2)
memory usage: 18.5 KB
```

In [5]:
```python
data.shape
```

Out[5]:
```
(214, 11)
```

In [6]:
```python
#data.head()
```

In [7]:
```python
# thống kê số lượng các lớp
data.groupby(10).count()[0]
```

Out[7]:
```
10
1    70
2    76
3    17
5    13
6     9
7    29
Name: 0, dtype: int64
```

In [8]:

```
# The columns that we will be making predictions with.
inputs = data.iloc[:,1:-1]
inputs.shape
```

Out[8]:

```
(214, 9)
```

In [9]:

```
inputs.head()
```

Out[9]:

	1	2	3	4	5	6	7	8	9
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0

In [10]:

```
# The column that we want to predict.
outputs = data[10]
outputs = np.array(outputs)
outputs.shape
```

Out[10]:

```
(214,)
```

In [11]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs, outputs,
                                                    test_size=0.30,
                                                    random_state=1)
```

Chúng ta không áp dụng AdaBoostClassifier với KNN vì KNeighborsClassifier không hỗ trợ sample_weight (mà trong AdaBoostClassifier cần)

AdaBoost

In [12]:

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
# mặc định là DecisionTreeClassifier() nên có thể không cần ghi
ml = DecisionTreeClassifier()
boosting = AdaBoostClassifier(n_estimators=100,
                         base_estimator=ml,
                         learning_rate=1)
```

In [13]:

```
# Train model
model_new = boosting.fit(X_train, y_train)
```

In [14]:

```
model_new.score(X_train, y_train)
```

Out[14]:

```
1.0
```

In [15]:

```
model_new.score(X_test, y_test)
```

Out[15]:

```
0.7384615384615385
```

In [16]:

```
# Kết luận: Overfitting
```

In [17]:

```
from sklearn.ensemble import RandomForestClassifier
ml_1 = RandomForestClassifier(n_estimators=100)
boosting_1 = AdaBoostClassifier(n_estimators=100,
                         base_estimator=ml_1,
                         learning_rate=0.1)
```

In [18]:

```
# Train model
boosting_1.fit(X_train, y_train)
```

Out[18]:

```
AdaBoostClassifier(algorithm='SAMME.R',
                   base_estimator=RandomForestClassifier(bootstrap=True,
                                                         class_weight=None,
                                                         criterion='gini',
                                                         max_depth=None,
                                                         max_features='auto',
                                                         max_leaf_nodes=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=100,
                                                         n_jobs=None,
                                                         oob_score=False,
                                                         random_state=None,
                                                         verbose=0,
                                                         warm_start=False),
                   learning_rate=0.1, n_estimators=100, random_state=None)
```

```
In [19]: boosting_1.score(X_train, y_train)
```

```
Out[19]: 1.0
```

```
In [20]: boosting_1.score(X_test, y_test)
```

```
Out[20]: 0.8153846153846154
```

```
In [32]: from sklearn.model_selection import cross_val_score
         scores1 = cross_val_score(boosting_1, inputs, outputs, cv=20)
         scores1
```

```
c:\program files\python36\lib\site-packages\sklearn\model_selection\_split.py:657: Warning: The least populated class in y has o
nly 9 members, which is too few. The minimum number of members in any class cannot be less than n_splits=20.
  % (min_groups, self.n_splits)), Warning)
```

```
Out[32]: array([0.69230769, 0.69230769, 0.92307692, 0.92307692, 0.76923077,
                0.84615385, 0.69230769, 0.76923077, 0.46153846, 0.63636364,
                0.9       , 0.8       , 1.        , 0.88888889, 0.33333333,
                0.66666667, 0.75      , 1.        , 1.        , 1.        ])
```

```
In [33]: display(np.mean(scores1),np.std(scores1))
```

```
0.7872241647241648

0.17651315097478404
```

```
In [23]: # Kết Luận: Vẫn overfitting nhưng có cải thiện hơn
         # Còn model nào tốt hơn không? Cho kết quả.
         # Thử áp dụng bài toán này với XGBoost.
```

XGBoost

```
In [24]: import xgboost as xgb
```

```
In [25]: xgb_model = xgb.XGBClassifier(random_state=42)
         xgb_model.fit(X_train, y_train)
```

```
Out[25]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                       colsample_bynode=1, colsample_bytree=1, gamma=0,
                       learning_rate=0.1, max_delta_step=0, max_depth=3,
                       min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
                       nthread=None, objective='multi:softprob', random_state=42,
                       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
                       silent=None, subsample=1, verbosity=1)
```

```
In [26]: xgb_model.score(X_train, y_train)
```

```
Out[26]: 1.0
```

```
In [27]: xgb_model.score(X_test, y_test)
```

```
Out[27]: 0.8307692307692308
```

```
In [30]: from sklearn.model_selection import cross_val_score
         scores2 = cross_val_score(xgb_model, inputs, outputs, cv=20)
         scores2
```

```
c:\program files\python36\lib\site-packages\sklearn\model_selection\_split.py:657: Warning: The least populated class in y has o
nly 9 members, which is too few. The minimum number of members in any class cannot be less than n_splits=20.
  % (min_groups, self.n_splits)), Warning)
```

```
Out[30]: array([0.69230769, 0.69230769, 0.92307692, 0.76923077, 0.84615385,
                0.76923077, 0.53846154, 0.84615385, 0.61538462, 0.72727273,
                1.        , 0.8       , 0.8       , 0.77777778, 0.11111111,
                0.77777778, 0.75      , 1.        , 1.        , 1.        ])
```

```
In [31]: display(np.mean(scores2),np.std(scores2))
```

```
0.7718123543123543

0.19677858024774542
```

```
In [ ]:
```