Chapter 8 - Exercise 3: Wine

Cho dữ liệu wine nằm trong tập tin wine.data.txt

(Xem chi tiết tại: http://archive.ics.uci.edu/ml/datasets/Wine)

Yêu cầu: đọc dữ liệu về, chuẩn hóa dữ liệu (nếu cần) và áp dụng thuật toán SVM để thực hiện việc dự đoán loại rượu dựa trên thông tin được cung cấp

1. Tạo X_train, X_test, y_train, y_test từ dữ liệu đọc được với tỷ lệ dữ liệu test là 0.3
2. Áp dụng thuật toán SVM
3. Tìm kết quả
4. Kiểm tra độ chính xác
5. Với X_new = [[13.71,5.65,2.45,20.5,95,1.68,.61,.52,1.06,7.7,.64,1.74,740], [12.29,1.61,2.21,20.4,103,1.1,1.02,.37,1.46,3.05,.906,1.82,870], [13.2,1.78,2.14,11.2,100,2.65,2.76,.26,1.28,4.38,1.05,3.4,1050]], thì y_new có kết quả ?
6. So sánh hiệu suất của 5 thuật toán: Logistic, Naive Bayes, SVM, RandomForestClassifier, DecisionTreeClassifier
7. Trực quan hóa kết quả

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

```
Mounted at /content/gdrive
```

In [ ]:
```
%cd '/content/gdrive/My Drive/LDS6_MachineLearning/practice_2020/Chapter8_SVM/'
```

```
/content/gdrive/My Drive/LDS6_MachineLearning/practice_2020/Chapter8_SVM
```

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

In [ ]:
```
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

In [ ]:
```
data = pd.read_csv('wine.data.txt', sep=',', header= None)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   0       178 non-null    int64  
 1   1       178 non-null    float64
 2   2       178 non-null    float64
 3   3       178 non-null    float64
 4   4       178 non-null    float64
 5   5       178 non-null    int64  
 6   6       178 non-null    float64
 7   7       178 non-null    float64
 8   8       178 non-null    float64
 9   9       178 non-null    float64
 10  10      178 non-null    float64
 11  11      178 non-null    float64
 12  12      178 non-null    float64
 13  13      178 non-null    int64  
dtypes: float64(11), int64(3)
memory usage: 19.6 KB
```

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

In [ ]:
```
X = data.iloc[:, 1:14]
y = data.iloc[:, 0]
```

In [ ]:
```
X.head()
```

Out[ ]:

	1	2	3	4	5	6	7	8	9	10	11	12	13
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735

In [ ]:
```
y.head()
```

Out[ ]:
```
0    1
1    1
2    1
3    1
4    1
Name: 0, dtype: int64
```

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3)
```

In [ ]:
```
clf = svm.SVC(kernel='linear')
clf.fit(X_train, y_train)
```

Out[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
y_pred = clf.predict(X_test)
```

In [ ]:
```
y_pred
```

Out[ ]:
```
array([2, 2, 2, 3, 1, 1, 3, 2, 2, 3, 1, 2, 2, 2, 2, 1, 3, 2, 3, 1, 1, 3,
       3, 2, 3, 2, 2, 1, 3, 2, 1, 1, 3, 2, 2, 2, 1, 1, 1, 3, 1, 1, 2, 1,
       2, 1, 2, 2, 2, 2, 1, 2, 3, 1])
```

In [ ]:
```
from sklearn.metrics import accuracy_score
print("Accuracy is ", accuracy_score(y_test,y_pred)*100,"%")
```

```
Accuracy is  98.14814814814815 %
```

In [ ]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[18  0  0]
 [ 0 23  0]
 [ 0  1 12]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        18
           2       0.96      1.00      0.98        23
           3       1.00      0.92      0.96        13

    accuracy                           0.98        54
   macro avg       0.99      0.97      0.98        54
weighted avg       0.98      0.98      0.98        54
```

In [ ]:
```
# Score of Training and Testing data
print("Training R^2 Score", clf.score(X_train, y_train))
print("Testing R^2 Score", clf.score(X_test, y_test))
```

```
Training R^2 Score 1.0
Testing R^2 Score 0.9814814814814815
```

Summary about the model:

- High accuracy: ~0.94
- High precision: ~0.94, High recall: ~0.95
- High training R^2 score and High testing score, nearly the same
- => The good model

In [ ]:
```
X_new = [[13.71,5.65,2.45,20.5,95,1.68,.61,.52,1.06,7.7,.64,1.74,740],
         [12.29,1.61,2.21,20.4,103,1.1,1.02,.37,1.46,3.05,.906,1.82,870],
        [13.2,1.78,2.14,11.2,100,2.65,2.76,.26,1.28,4.38,1.05,3.4,1050]]
y_new = clf.predict(X_new)
y_new
```

Out[ ]:
```
array([3, 2, 1])
```

In [ ]:
```
# Tính độ chính xác mdoel theo:
# Logistic, Naive Bayes, SVM, RandomForestClassifier, DecisionTreeClassifier
# Khi dùng KNN thì cần chọn k phù hợp trước
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
import datetime
from sklearn.model_selection import cross_val_score

models = [
    LogisticRegression(max_iter=200),
    GaussianNB(),
    SVC(kernel='linear'),
    RandomForestClassifier(n_estimators=200),
    DecisionTreeClassifier()
]
CV = 10
entries = []
i=0
for model in models:
    scores_train = []
    scores_test = []
    times = []
    abs_scores = []
    for j in range(CV):
        X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                            test_size=0.3)
        t1 = datetime.datetime.now()
        model_name = model.__class__.__name__
        model.fit(X_train,y_train)
        t2 = datetime.datetime.now()
        d = round((t2 - t1).microseconds/1000,1) # => miliseconds
        score_train = model.score(X_train,y_train)
        score_test = model.score(X_test,y_test)
        abs_score = abs(score_train - score_test)

        scores_train.append(score_train)
        scores_test.append(score_test)
        abs_scores.append(abs_score)
        times.append(d)

    print(model.__class__.__name__, scores_test)
    entries.append([model_name, np.array(scores_train).mean(),
                    np.array(scores_test).mean(), np.array(abs_scores).mean(),
                    np.array(times).mean()])
    i += 1
cv_df = pd.DataFrame(entries,
                     columns=['model_name', 'score_train_mean',
                              'score_test_mean', 'abs|score|', 'time_mean'])
```

In [ ]:
```
cv_df
```

Out[ ]:

	model_name	score_train_mean	score_test_mean	abs|score|	time_mean
0	LogisticRegression	0.959677	0.944444	0.015233	46.96
1	GaussianNB	0.959677	1.000000	0.040323	1.44
2	SVC	1.000000	0.981481	0.018519	91.18
3	RandomForestClassifier	1.000000	1.000000	0.000000	265.96
4	DecisionTreeClassifier	1.000000	0.870370	0.129630	1.24

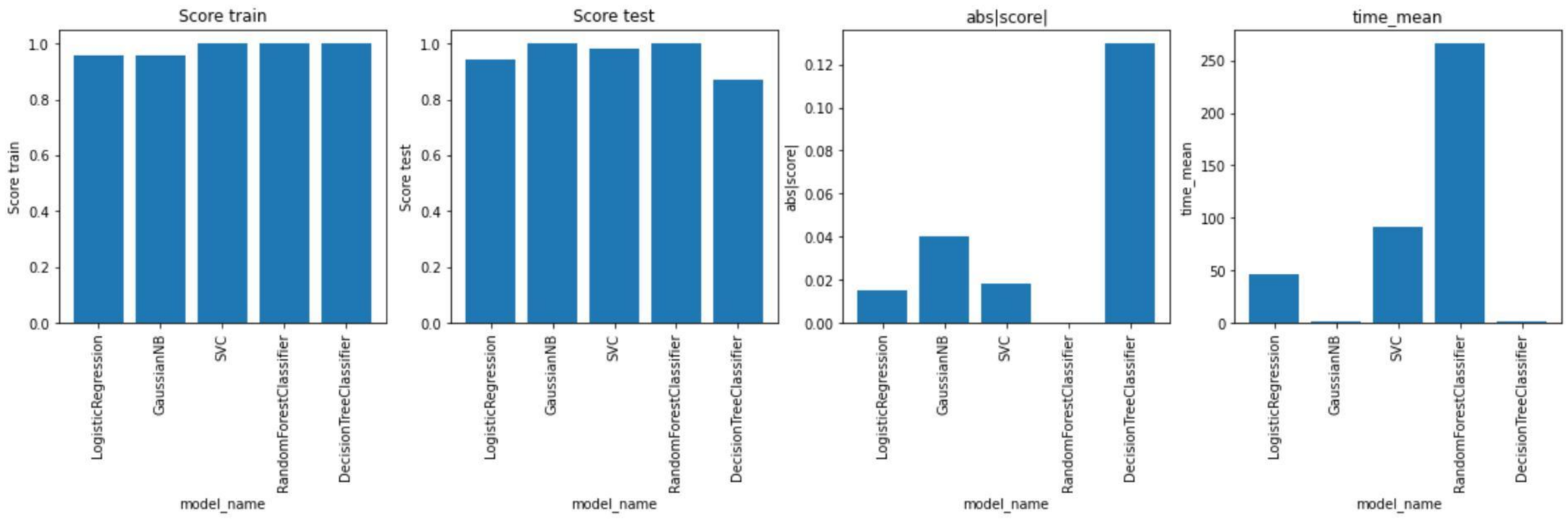
In [ ]:
```
plt.figure(figsize=(20, 4))
plt.subplot(1, 4, 1)
plt.bar(cv_df['model_name'],cv_df['score_train_mean'])
plt.xlabel('model_name')
plt.ylabel('Score train')
plt.xticks(rotation='vertical')
plt.title("Score train")

plt.subplot(1, 4, 2)
plt.bar(cv_df['model_name'],cv_df['score_test_mean'])
plt.xlabel('model_name')
plt.ylabel('Score test')
plt.xticks(rotation='vertical')
plt.title("Score test")

plt.subplot(1, 4, 3)
plt.bar(cv_df['model_name'],cv_df['abs|score|'])
plt.xlabel('model_name')
plt.ylabel('abs|score|')
plt.xticks(rotation='vertical')
plt.title("abs|score|")
```

```
plt.subplot(1, 4, 4)
plt.bar(cv_df['model_name'],cv_df['time_mean'])
plt.xlabel('model_name')
plt.ylabel('time_mean')
plt.xticks(rotation='vertical')
plt.title("time_mean")

plt.show()
```



In [ ]:
```
# nhan xet
# chon model nao?
# build model do tu A_Z
```