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 (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]], t
   hì y_new có kết quả ?
```

6. So sánh hiệu suất của 4 thuật toán: RandomForestClassifier, SVC, GaussianNB, LogisticRegression
7. Trực quan hóa kết quả

In [1]:

```
# from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
```

In [2]:

```
# %cd '/content/gdrive/My Drive/LDS6_MachineLearning/practice/Chapter8_SVM/'
```

In [3]:

```
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 [4]:

```
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

In [5]:
```
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):
0     178 non-null int64
1     178 non-null float64
2     178 non-null float64
3     178 non-null float64
4     178 non-null float64
5     178 non-null int64
6     178 non-null float64
7     178 non-null float64
8     178 non-null float64
9     178 non-null float64
10    178 non-null float64
11    178 non-null float64
12    178 non-null float64
13    178 non-null int64
dtypes: float64(11), int64(3)
memory usage: 19.6 KB
```

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

In [7]:
```
X = data.iloc[:, 1:14]
y = data.iloc[:, 0]
```

In [8]:
```
X.head()
```

Out[8]:

	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 [9]:
```
y.head()
```

Out[9]:
```
0    1
1    1
2    1
3    1
4    1
Name: 0, dtype: int64
```

In [10]:
```
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 [11]: clf = svm.SVC(kernel='linear')
         clf.fit(X_train, y_train)
```

```
Out[11]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
             decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
             kernel='linear', max_iter=-1, probability=False, random_state=None,
             shrinking=True, tol=0.001, verbose=False)
```

```
In [12]: y_pred = clf.predict(X_test)
```

```
In [13]: y_pred
```

```
Out[13]: array([2, 2, 2, 1, 1, 3, 2, 3, 1, 3, 2, 3, 2, 2, 1, 3, 3, 1, 3, 1, 2, 1,
                1, 1, 3, 3, 2, 1, 2, 3, 1, 3, 1, 1, 2, 3, 1, 1, 2, 2, 2, 2, 2, 3,
                3, 1, 2, 2, 2, 2, 1, 1, 1, 2], dtype=int64)
```

```
In [14]: from sklearn.metrics import accuracy_score
         print("Accuracy is ", accuracy_score(y_test,y_pred)*100,"%")
```

```
Accuracy is  94.44444444444444 %
```

```
In [15]: from sklearn.metrics import classification_report, confusion_matrix
         print(confusion_matrix(y_test,y_pred))
         print(classification_report(y_test,y_pred))
```

```
[[19  2  0]
 [ 0 19  1]
 [ 0  0 13]]
              precision    recall  f1-score   support

           1       1.00      0.90      0.95        21
           2       0.90      0.95      0.93        20
           3       0.93      1.00      0.96        13

    accuracy                           0.94        54
   macro avg       0.94      0.95      0.95        54
weighted avg       0.95      0.94      0.94        54
```

```
In [16]: # 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 0.9919354838709677
Testing R^2 Score 0.9444444444444444
```

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 [17]:
```
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[17]:
```
array([3, 2, 1], dtype=int64)
```

In [18]:
```
# Tính độ chính xác theo: Logistic, Naive Bayes, SVM, KNN
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import cross_val_score
models = [
    RandomForestClassifier(n_estimators=200),
    SVC(kernel='linear'),
    GaussianNB(),
    DecisionTreeClassifier()
]
CV = 5
cv_df = pd.DataFrame(index=range(CV * len(models)))
entries = []
i=0
for model in models:
    scores = []
    for j in range(CV):
        model_name = model.__class__.__name__
        model.fit(X,y)
        score = model.score(X,y)
        scores.append(score)
    print(model.__class__.__name__, scores)
    entries.append([model_name, np.array(scores).mean()])
    i += 1
cv_df = pd.DataFrame(entries, columns=['model_name', 'score_mean'])
```

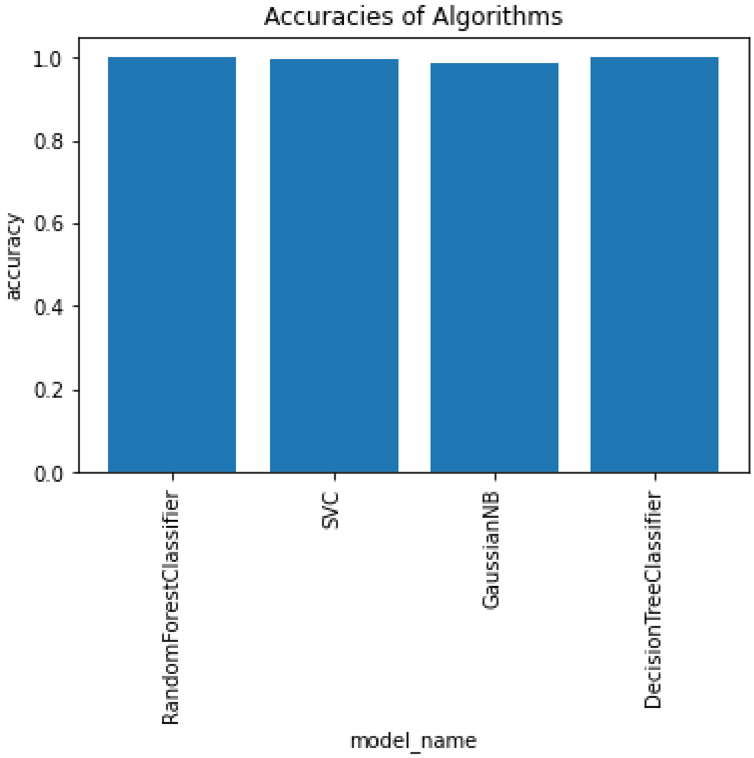
```
RandomForestClassifier [1.0, 1.0, 1.0, 1.0, 1.0]
SVC [0.9943820224719101, 0.9943820224719101, 0.9943820224719101, 0.994382022471
9101, 0.9943820224719101]
GaussianNB [0.9887640449438202, 0.9887640449438202, 0.9887640449438202, 0.98876
40449438202, 0.9887640449438202]
DecisionTreeClassifier [1.0, 1.0, 1.0, 1.0, 1.0]
```

In [19]:
```
cv_df
```

Out[19]:

	model_name	score_mean
0	RandomForestClassifier	1.000000
1	SVC	0.994382
2	GaussianNB	0.988764
3	DecisionTreeClassifier	1.000000

```
In [20]: plt.bar(cv_df['model_name'],cv_df['score_mean'])
         plt.xlabel('model_name')
         plt.ylabel('accuracy')
         plt.xticks(rotation='vertical')
         plt.title("Accuracies of Algorithms")
         plt.show()
```


Accuracies of Algorithms
accuracy
RandomForestClassifier
SVC
GaussianNB
DecisionTreeClassifier
model_name