Ex 3: Skin Color

Bộ dữ liệu phân loại da (Skin Segmentation) được tạo thành từ 3 không gian màu B, G, R. Dữ liệu Skin và Nonskin được tạo ra bằng cách sử dụng kết cấu da từ hình ảnh khuôn mặt với sự đa dạng về độ tuổi, giới tính,...

Có (245057 * 4) sample với 3 cột đầu là B,G,R (x1,x2, và x3 features), cột thứ tư là class labels (y).

Cho dữ liệu skin nằm trong tập tin Skin_NonSkin.txt

(xem chi tiết tại: https://archive.ics.uci.edu/ml/datasets/skin+segmentation (https://archive.ics.uci.edu/ml/datasets/skin+segmentation)) Yêu cầu: đọc dữ liệu về, chuẩn hóa dữ liệu (nếu cần) và áp dụng thuật toán SVM để thực hiện việc dự đoán có Skin hay NonSkin 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 = [[76, 86, 126], [170, 170, 120]], thì y_new có kết quả ?

In [1]:
```
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 [2]:
```
import datetime
x1 = datetime.datetime.now()
print(x1)
```

```
2019-08-27 08:56:42.923106
```

In [3]:
```
data = pd.read_csv('Skin_NonSkin.txt', sep='\t', header= None)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 245057 entries, 0 to 245056
Data columns (total 4 columns):
0    245057 non-null int64
1    245057 non-null int64
2    245057 non-null int64
3    245057 non-null int64
dtypes: int64(4)
memory usage: 7.5 MB
```

In [4]:
```
data.head()
```

Out[4]:

	0	1	2	3
0	74	85	123	1
1	73	84	122	1
2	72	83	121	1
3	70	81	119	1
4	70	81	119	1

In [5]:
```
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
```

In [6]:
```
X.head()
```

Out[6]:

	0	1	2
0	74	85	123
1	73	84	122
2	72	83	121
3	70	81	119
4	70	81	119

In [7]:
```
y.head()
```

Out[7]:
```
0    1
1    1
2    1
3    1
4    1
Name: 3, dtype: int64
```

In [8]:
```
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) # 70% training and 30% test
```

In [9]:
```
clf = svm.SVC(gamma=0.001, C=100)
clf.fit(X_train, y_train)
```

Out[9]:
```
SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In [10]:
```
y_pred = clf.predict(X_test)
```

In [11]:
```
y_pred
```

Out[11]:
```
array([2, 1, 2, ..., 2, 2, 2], dtype=int64)
```

In [12]:
```
from sklearn.metrics import accuracy_score
print("Accuracy is ", accuracy_score(y_test,y_pred)*100,"%")
```

```
Accuracy is  99.96191408906662 %
```

In [13]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[15196     3]
 [   25 58294]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00     15199
           2       1.00      1.00      1.00     58319

   micro avg       1.00      1.00      1.00     73518
   macro avg       1.00      1.00      1.00     73518
weighted avg       1.00      1.00      1.00     73518
```

In [14]:
```
from sklearn import metrics
# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:",metrics.precision_score(y_test, y_pred))

# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Precision: 0.9983575323566126
Recall: 0.9998026185933285
```

In [15]:
```
X_new = [[76, 86, 126], [170, 170, 120]]
y_new = clf.predict(X_new)
y_new
```

Out[15]:
```
array([1, 2], dtype=int64)
```

In [16]:
```
x2 = datetime.datetime.now()
print(x2)
```

```
2019-08-27 08:56:47.964366
```

In [17]:
```
d = x2 - x1
print(d)
```

```
0:00:05.041260
```