多媒体技术大作业 1

2018302080152_网安 1 班_范圣悦

一、 问题描述

在给定的图片库里，以颜色直方图为依据，用最近邻方法或者其它方法找到某张图片的类别。随机抽取一张图片，利用模板匹配方法，在图片库中找出 100 幅最相似的图片，计算查全率和查准率。

二、 解题思路

1、 利用 sklearn 的随机森林分类器，以颜色直方图为依据，训练数据集中的图片得到模型，利用模型预测某张图片的类别。

2、 假设源图片 img 真实类别为 X，则在所有图片中，找到模型预测出类别为 X 的所有图片。

3、 为第二步找到的所有图片计算颜色直方图，利用 calchist()函数找到与源图片 img 相似度最接近的五张照片。

三、 实验环境

电脑配置：Intel(R) Core(TM) i5-8300H CPU @2.30GHz 2.30GHz,16G 内存

操作系统：Windows10 家庭中文版[版本 10.0.17763.1217]

开发工具：Pycharm 2020.1 x64

四、 方法流程说明

1．读取训练集和测试集。

X_train 列表保存训练集图像名称，y_train 列表保存训练集图像类标；

X_test 列表保存测试集图像名称，y_test 列表保存测试集图像类标；

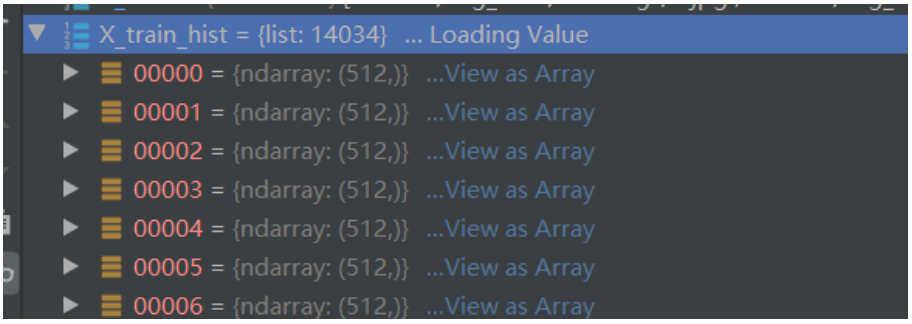
其中训练集为 seg_train 文件夹，包含 6 个不同类别['buildings', 'forest', 'glacier', 'mountain', 'sea', 'street']的 14034 张图片。

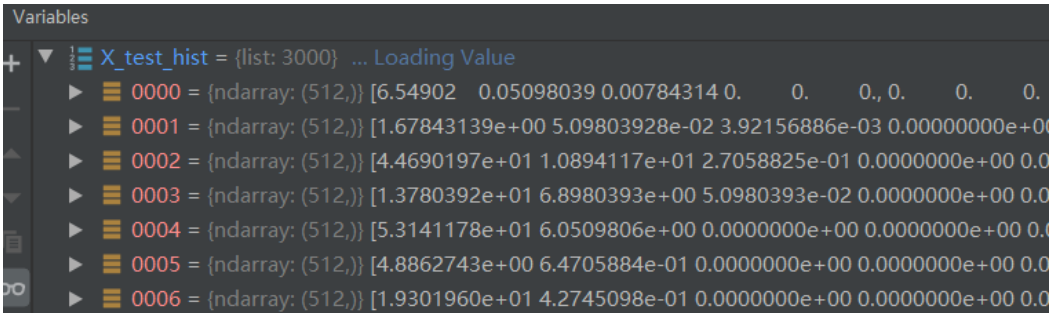
测试集为 seg_test 文件夹，包含 6 个不同类别['buildings', 'forest', 'glacier', 'mountain', 'sea', 'street']的 3000 张图片。

```
print(len(X_train), len(X_test), len(y_train), len(y_test))  # 查看训练集和测试集的大小
```

```
14034 3000 14034 3000
```

2. 分别计算训练集和测试集的颜色直方图，经过测试发现 RGB 三通道直方图效果略优于 HSV 模型直方图，这里采用 RGB 三通道颜色直方图。计算后分别用 X_trian_hist 和 X_test_hist 保存得到的直方图。





3. 使用 sklearn 库的随机森林分类器 RandomForestClassifier，设置恰当的参数，保存模型到本地，以便下一次直接加载保存好的模型。

4. 利用 sklearn. metrics 方法计算算法的各种评价指标，分别计算其准确率 accuracy、查准率 precision 和查全率 recall，结果保留四位小数，

如下图所示：

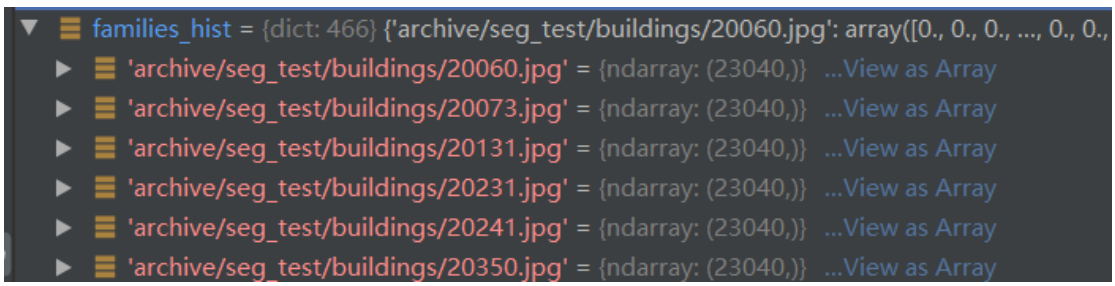
```
准确率accuracy：0.6157
查准率Precision：0.6143
查全率reall：0.6086
              precision    recall  f1-score   support

   buildings       0.43      0.31      0.36       437
      forest       0.87      0.74      0.80       474
     glacier       0.65      0.74      0.69       553
    mountain       0.59      0.58      0.58       525
         sea       0.60      0.55      0.57       510
      street       0.55      0.74      0.63       501

 avg / total       0.62      0.62      0.61      3000
```

5. 找出原图片的真实类别 X，利用随机森林分类器找到所有图片的预测分类与原图片真实类别 X 一样的图片，保存这些同类别图片的名称到列表 families_nams 中。

6. 计算第五步得到的所有同分类图片的颜色直方图，经过测试发现 RGB 三通道直方图效果差于HSV模型直方图，这里采用HSV模型颜色直方图。计算得到的直方图保存在 families_hist 中。如下图所示：



7. openCV 库提供了直方图比较的方法，对输入的两张图像计算得到直方图 H1 与 H2，归一化到相同的尺度空间，然后可以通过计算 H1 与 H2 的之间的距离得到两个直方图的相似程度，进而比较图像本身的相似程度。将同类别的所有图片颜色直方图与源图片 img 的颜色直方图一一比较，有 4 种比较方法可供选择：

① 相关性比较（method=cv2.HISTCMP_CORREL）值越大，相关度越高，最大值为 1，最小值为 0

$$d(H_1,H_2)=\frac{\sum_I(H_1(I)-\bar{H_1})(H_2(I)-\bar{H_2})}{\sqrt{\sum_I(H_1(I)-\bar{H_1})^2\sum_I(H_2(I)-\bar{H_2})^2}}$$

② 卡方比较(method=cv2.HISTCMP_CHISQR 值越小，相关度越高，最大值无上界，最小值 0

$$d(H_1,H_2)=\sum_I\frac{(H_1(I)-H_2(I))^2}{H_1(I)}$$

③ 巴氏距离比较(method=cv2.HISTCMP_BHATTACHARYYA）值越小，相关度越高，最大值为 1，最小值为 0

$$d(H_1,H_2)=\sqrt{1-\frac{1}{\sqrt{\bar{H_1}\bar{H_2}N^2}}\sum_I\sqrt{H_1(I)\cdot H_2(I)}}$$

④ 交集法比较(method=CV_COMP_INTERSECT)，取两个直方图每个相同位置的值的最小值，然后求和。值越大越相似。

$$d(H_1,H_2)=\sum_I\min(H_1(I),H_2(I))$$

8. 这里选择巴氏距离比较方法，计算出所有图片与源图片的相似度后，按照从小到大的顺序排序，选择前五副作为与源图片最接近的五幅图片。

从六个不同分类中分别选择一张作为源图片，程序结果如下图所示：

（1）buildings

```
图片archive/seg_test/buildings/20776.jpg的预测分类是:buildings；真实分类是:buildings
与上述图片最接近的五幅图片分别是：
('archive/seg_test/buildings/20776.jpg', '0.0')
('archive/seg_test/buildings/22496.jpg', '0.5434956450614551')
('archive/seg_test/street/22109.jpg', '0.5917808171874183')
('archive/seg_test/buildings/21386.jpg', '0.591826230064269')
('archive/seg_test/mountain/24006.jpg', '0.5997455707300885')
```

源图片：

最相似的五张图片：

图片编号	20776	22496	22109	21386	24006
图片					

类别分别是 buildings、buildings、street、buildings、mountain

（2）forest

```
图片archive/seg_test/forest/22293.jpg的预测分类是:forest；真实分类是:forest
与上述图片最接近的五幅图片分别是：
('archive/seg_test/forest/22293.jpg', '0.0')
('archive/seg_test/forest/21126.jpg', '0.3802123952821678')
('archive/seg_test/forest/21459.jpg', '0.3998267127753898')
('archive/seg_test/forest/23241.jpg', '0.4014509777347002')
('archive/seg_test/forest/21317.jpg', '0.4015917424099624')
```

源图片：

最相似的五幅图片：

图片编号	22293	21126	21459	23241	21317
图片					

类别分别是 forest、forest、forest、forest、forest

（3）glacier

```
图片archive/seg_test/glacier/23548.jpg的预测分类是:glacier；真实分类是:glacier
与上述图片最接近的五幅图片分别是：
('archive/seg_test/glacier/23548.jpg', '0.0')
('archive/seg_test/glacier/20275.jpg', '0.3892523552192112')
('archive/seg_test/glacier/20243.jpg', '0.3965618322425178')
('archive/seg_test/mountain/22765.jpg', '0.40378014355276515')
('archive/seg_test/sea/24235.jpg', '0.41421125644944945')
```

源图片：

最相似的五幅图片：

图片编号	23548	20275	20243	22765	24235
图片					

类别分别是 glacier、glacier、glacier、mountain、sea

（4）mountain

```
图片archive/seg_test/mountain/21236.jpg的预测分类是:mountain；真实分类是:mountain
与上述图片最接近的五幅图片分别是：
('archive/seg_test/mountain/21236.jpg', '0.0')
('archive/seg_test/glacier/21469.jpg', '0.5649116987443011')
('archive/seg_test/mountain/23807.jpg', '0.5992134980494934')
('archive/seg_test/mountain/23617.jpg', '0.601289884718229')
('archive/seg_test/mountain/22444.jpg', '0.6056020491771947')
```

源图片：

最接近的五张图片：

图片编号	21236	21469	23807	23617	22444
图片					

类别分别是 mountain、glacier、mountain、mountain、mountain

（5）sea

```
图片archive/seg_test/sea/20389.jpg的预测分类是:sea；真实分类是:sea
与上述图片最接近的五幅图片分别是：
('archive/seg_test/sea/20389.jpg', '0.0')
('archive/seg_test/sea/20476.jpg', '0.5179102673378985')
('archive/seg_test/sea/23623.jpg', '0.6166982701049644')
('archive/seg_test/glacier/23189.jpg', '0.6286842044424025')
('archive/seg_test/sea/22107.jpg', '0.6307096592991477')
```

源图片：

最接近的五张图片：

图片编号	20389	20476	23623	23189	22107
图片					

类别分别是：sea、sea、sea、glacier、sea

（6）street

```
图片archive/seg_test/street/24332.jpg的预测分类是:street；真实分类是:street
与上述图片最接近的五幅图片分别是：
('archive/seg_test/street/24332.jpg', '0.0')
('archive/seg_test/buildings/21159.jpg', '0.43193500427562975')
('archive/seg_test/street/20359.jpg', '0.4406987938675345')
('archive/seg_test/street/22406.jpg', '0.44596128147332226')
('archive/seg_test/buildings/23543.jpg', '0.4932377115614276')
```

源图片：

最接近的五张图片：

图片编号	24332	21159	20359	22406	23543
图片					

类别分别是：street、buildings、street、street、buildings

五、 代码与代码说明

1、读取训练集和测试集：

主函数调用：

```
84  X_train = []  # 训练集图像名称
85  y_train = []  # 训练集图像分类类标
86  X_test = []   # 测试集图像名称
87  y_test = []   # 测试集图像分类类标
88  classifications = ['buildings', 'forest', 'glacier', 'mountain', 'sea', 'street'] # 图片的所有分类
89  read_train(X_train, y_train, classifications) # 读取训练集
90  read_test(X_test, y_test, classifications)  # 读取测试集
91  print(len(X_train), len(X_test), len(y_train), len(y_test))  # 查看训练集和测试集的大小
```

具体函数实现：

```
10  #读取训练集
11  def read_train(X, Y, classifications):
12      for i in classifications:
13          # 遍历文件夹，读取图片
14          for f in os.listdir("archive/seg_train/%s" % i):
15              # 获取图像名称
16              X.append("archive/seg_train/" + str(i) + "/" + str(f))
17              # 获取图像类标即为文件夹名称
18              Y.append(i)
19      X = np.array(X)
20      Y = np.array(Y)
21  #读取测试集
22  def read_test(X, Y, classifications):
23      for i in classifications:
24          # 遍历文件夹，读取图片
25          for f in os.listdir("archive/seg_test/%s" % i):
26              # 获取图像名称
27              X.append("archive/seg_test/" + str(i) + "/" + str(f))
28              # 获取图像类标即为文件夹名称
29              Y.append(i)
30      X = np.array(X)
31      Y = np.array(Y)
```

read_train 和 read_test 函数分别用于读取数据集和训练集，其中 X 数组用于保存图片的名称（例如’archive/seg_test/sea/20389.jpg’），Y 数组保存类别标签。

2、计算训练集和测试集的 RGB 颜色直方图：

主函数调用：

```
92      X_train_hist = []   # 保存训练集的颜色直方图
93      X_test_hist = []    # 存测试集的颜色直方图
94      # 分别计算训练集和测试集的颜色直方图
95      data_hist(X_train, X_train_hist)
96      data_hist(X_test, X_test_hist)
```

具体函数实现：

```
33  #计算颜色直方图
34  def data_hist(X, X_hist):
35      for i in X:
36          # 读取图像
37          image = cv2.imread(i)
38          # 图像像素大小一致
39          img = cv2.resize(image, (256, 256), interpolation=cv2.INTER_CUBIC)
40          # 计算图像直方图并存储至X数组
41          hist = cv2.calcHist([img], [0, 1, 2], None, [8, 8, 8], [0.0, 255.0, 0.0, 255.0, 0.0, 255.0])
42          X_hist.append(((hist / 255).flatten())) #利用.flatten函数返回一维数组
```

data_hist 函数用于计算图片的颜色直方图，函数参数中，X 代表保存图片的列表，X_hist 为计算得到的颜色直方图。首先利用 cv2.resize 函数将图片全部调整到同样大小，随后利用 cv2.calcHist 函数计算 RGB 三个通道颜色直方图，最后利用 numpy.flatten 函数得到一维数组。

利用 data_hist，先计算出测试集和训练集图片的颜色直方图，分别保存在 X_train_hist 列表和 X_test_hist 列表中。

3、使用随机森林分类器

```
98      try:
99          with open("model.pkl", "rb") as f:  # 若模型已经存在，加载已保存的模型
100             model = pickle.load(f)
101     except Exception:   # 若模型不存在，重新训练
102         model = RandomForestClassifier(n_jobs=-1, n_estimators=350, max_depth=11,
103                                        oob_score=1, random_state=1).fit(X_train_hist, y_train)
104         with open("model.pkl", "wb") as f:  # 保存模型
105             pickle.dump(model, f)
```

首先检测模型是否已经存在，若存在，则加载已经保存的模型，否则重新训练。利用 sklearn 库中的 RandomForestClassifier 函数训练模型。

RandomForestClassifier 的参数解释如下：

n_jobs=-1：代表运用所有的处理器进行训练；

n_estimators ：随机森林中树的数量；

max_depth：树的最大深度；

oob_score：使用袋外（out-of-bag）样本估计准确度；

random_state：随机数种子。

以训练集的 RGB 颜色直方图为输入数据，训练集的分类标签为标签对模型进行训练，训练完成后，把模型保存为 model.pkl。

4、对测试集进行预测和指标评价

```
106     predictions_labels = model.predict(X_test_hist)  # 保存预测结果
107     # 输出算法评价指标
108     print('准确率accuracy: {:.4f}'.format(metrics.accuracy_score(y_test, predictions_labels)))
109     print('查准率Precision: {:.4f}'.format(metrics.precision_score(y_test, predictions_labels, average='macro')))
110     print('查全率reall: {:.4f}'.format(metrics.recall_score(y_test, predictions_labels, average='macro')))
111     print(classification_report(y_test, predictions_labels))
```

对测试集图片的颜色直方图进行预测，得到测试集中每一张图片的分类标签，将其保存在 predictions_labels 列表中。

调用 sklearn.metrics 方法，分别计算准确率 accuracy、查准率 precision、查全率 recall，均保留四位小数。最后用 classification_report 函数用于显示主要分类指标的文本报告。

5、找出将要进行对比的图片 img，记录其预测类别和真实类别

```
number = 2035   # 该副图片img的编号，从0~2999
src_img = X_test[number]    # img的名称/路径
# 找出源图片的真实分类
for i in classifications:
    if i in src_img:
        src_label = i
print('图片{}的预测分类是:'.format(src_img)+str(predictions_labels[number])+'；真实分类是:'+str(src_label))
```

6、找出测试集所有图片中与 img 同类的图片

```
# 找到与img真实分类同一分类下的所有图片
families = []
for i in range(0, len(X_test)):
    if predictions_labels[i] == src_label:
        families.append(i)
# 保存同一分类所有图片的名称/路径
families_names = []
for i in families:
    families_names.append(X_test[i])
```

families 列表找出与 img 真实分类同类别的所有图片，famalies_names 保存同一分类所有图片的名称。

7、计算 famalies_names 中所有图片的 HSV 颜色直方图

主函数调用：

```
families_hist = {}  # 保存同一分类所有图片的颜色直方图
calchist_HSV(families_names, families_hist)  # 计算同一分类所有图片的颜色直方图
```

具体函数实现：

```
#计算HSV直方图
def calchist_HSV(data, data_hist):
    for i, filename in enumerate(data):
        image = cv2.imread(filename)
        hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
        hist1 = cv2.calcHist([hsv], [0, 1], None, [90, 256], [0, 180, 0, 256])
        hist1 = cv2.normalize(hist1, hist1).flatten()
        data_hist[filename] = hist1
```

families_hist 为一个字典，每一个键表示与 img 同分类的图片名称，每一个键的值表示对应图片的 HSV 颜色直方图。

calchist_HSV 函数用于计算 HSV 直方图。在比较图像相似度时，HSV 颜色空间的效果比 RGB 颜色空间更优，因此首先利用 cv2.cvtColor 函数转换图像的颜色空间，将 RGB 颜色空间转换为 HSV 颜色空间。随后利用 calcHist 函数计算颜色直方图，再将得到数据归一化处理（归一化是为了后面数据处理的方便，其次是保证程序运行时收敛加快），最后把键值添加到字典中。

8、基于直方图进行比较，找出最接近的五幅图片

主函数调用：

```
135     hist_compare(src_img, families_hist)  # 与img的直方图进行比较，找出最接近的五幅图片
```

具体函数实现：

```
63  #比较颜色直方图
64  def hist_compare(src, index):
65      result = {}  # 保存相似度计算结果
66      compareFileName = src # 需要进行相似比较的图像
67      for (k, hist) in index.items():
68          # 使用巴氏距离方法，该值越小，则两幅图像越相似
69          d = cv2.compareHist(index[compareFileName], hist, cv2.HISTCMP_BHATTACHARYYA)
70          result[k] = str(d)
71      # 排序相似度结果，从小到大
72      result = sorted(result.items(), key=lambda x: x[1])
73      # 选择最相似的前五幅
74      result = result[:5]
75      print("与上述图片最接近的五幅图片分别是：")
76      for i in result:
77          print(i)
```

使用 cv2.compareHist 函数进行直方图比较，compareHist 函数的第三个参数代表比较的方法，由于巴氏距离方法比较准确，所以这里采用巴氏距离。将比较后得到的相似度结果存放在 result 中。

把所有图片都与 img 比较完后，对 result 中的相似度结果进行排序，按照从小到大的顺序。最后截取前五副，打印输出。

六、 结论与分析

1、 分析随机森林分类器模型的查准率、查全率优化

① 不同颜色空间对模型效果的影响：

分别测试计算直方图时，选择 RGB 颜色空间和 HSV 颜色空间对模型效果的影响，得到结果如下：

	RGB	HSV
accuracy	0.6157	0.5985
precision	0.6143	0.5897
recall	0.6086	0.5914

可以发现，在对数据进行特征提取的时候，选用 RGB 颜色空间比选用 HSV 的效果略好一些。

② calcHist 函数中参数对效果的影响：

计算 RGB 直方图时，calcHist 中的参数 histSize，表示这个直方图分成多少份（即 bins，多少个直方柱）。分别测试计算直方图时，选择不同的 bins 数对模型效果的影响，得到结果如下（bins 数太大会内存爆炸无法计算）：

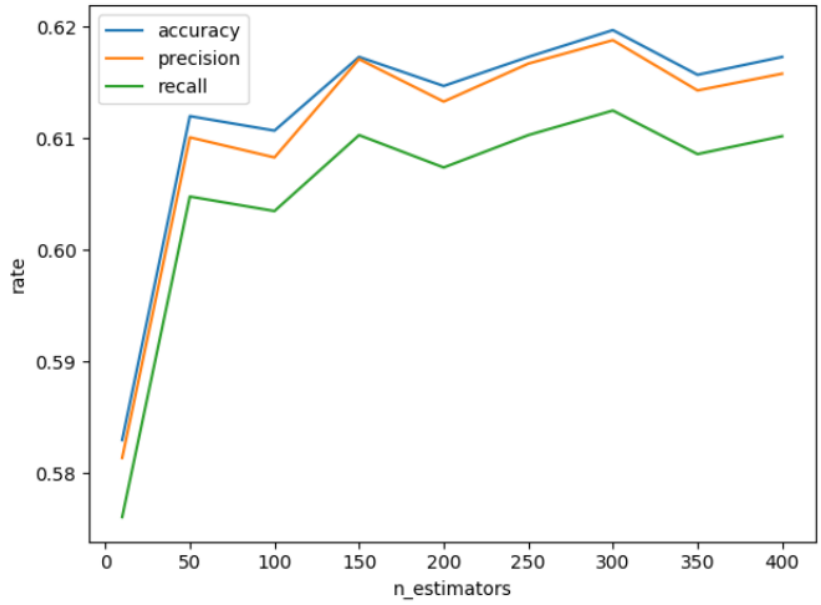
bins数	8	16	64
accuracy	0.6157	0.6097	0.5986
precision	0.6143	0.6032	0.5977
recall	0.6086	0.6011	0.5926

计算 RGB 直方图时，calcHist 中的第二个参数，表示计算直方图的通道。分别测试计算直方图时，选择不同的通道数对模型效果的影响，得到结果如下：

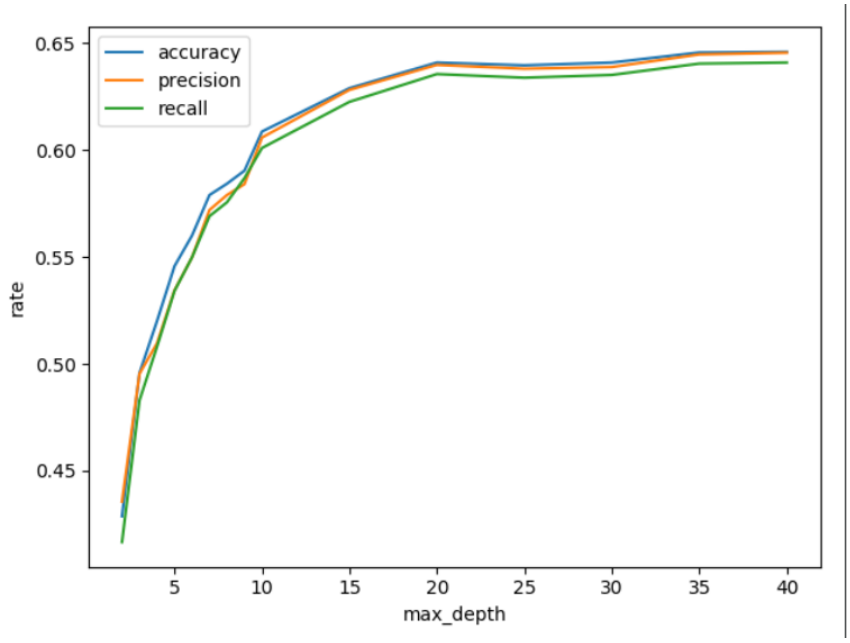
通道数	3	2	1
accuracy	0.6157	0.588	0.4997
precision	0.6143	0.5815	0.4859
recall	0.6086	0.5818	0.4963

③ 随机森林分类器参数对效果的影响：

改变 n_estimators（随机森林中树的数量），多次测试得到改变结果如下图所示：



改变 max_depth（树的最大深度），多次测试得到改变结果如下图所示：



2、 实验结论

由上面的参数分析可知：

① 对数据进行特征提取的时候，选用 RGB 颜色空间比选用 HSV 的效果略好一些；

② 计算 RGB 直方图时，bins 取 8 时效果最优；

③ 随机森林分类器中树的数量在 300 左右时最优，但 200 以上变化不大；

④ 随机森林分类器中树的深度越深，效果越好，但 20 以上变化不大；

同时，通过直方图比较分析得到以下结论：

① 得到的最相似图片可能与原图片分类并不同。

原因是：直方图表示颜色在图像中所占的比例，主要考虑颜色，不考虑形状，无法准确描述图像内容，存在误差。

② 由上述报告得到五幅最接近图片可知，他们的颜色比例都十分相近，但形状大小并不相似。

③ 一些特定分类的图片，如海和冰川、街道和建筑，他们的颜色非常接近，因此查找海的相似图片，可能得到冰川，查找街道的相似图片，可能得到建筑；而另一些图片，如森林，由于其独特的绿色在其余分类中比较少见，通常找相似时能够找到同一类别。

本次项目的 GitHub 地址：https://github.com/fshooty379/2020multi-media