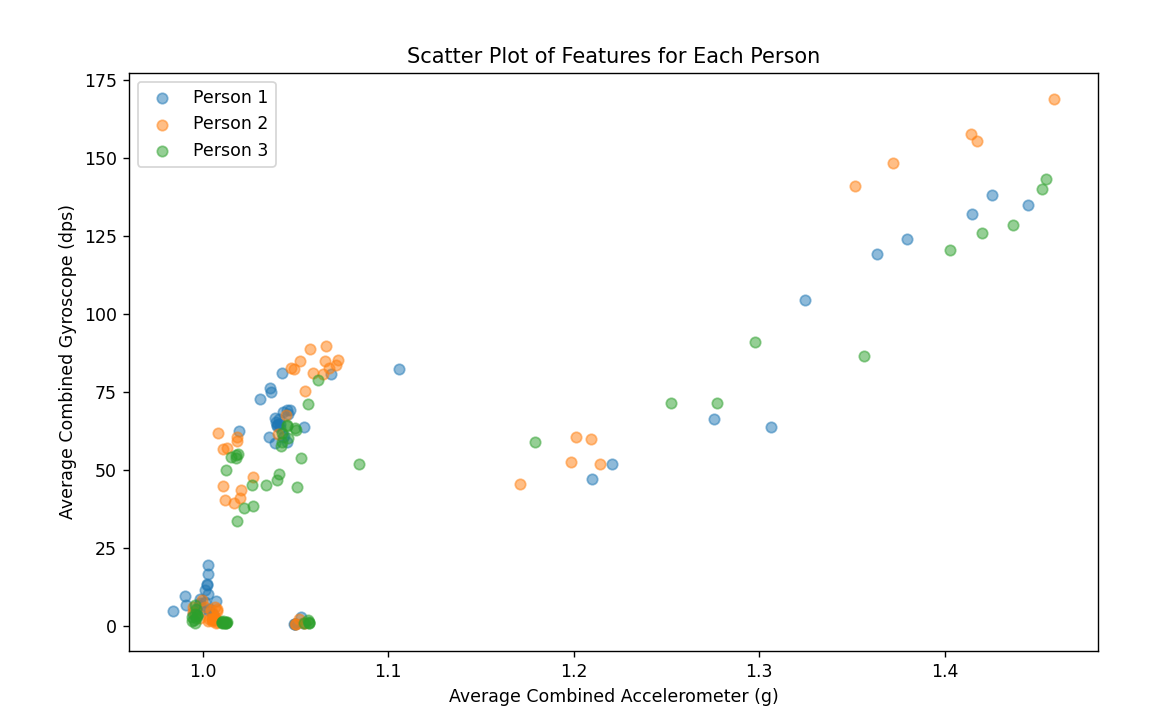
# 第一问

每组数据记录其线加速度和角加速度。所以表格每行有两个三维坐标数据，将每个三维数据变为一个特征数字，一行数据恰好为一个点坐标。分别对每个表格文件的每行求均值，大致加速度分布如下图：



其中求相似度直接采用了余弦相似度节省计算，在寻找最优化分组时，由于代码限制，我用了贪心来简化了代码复杂度。

import os  
import matplotlib.pyplot as plt  
import pandas as pd  
import pulp  
from scipy.stats import t  
import numpy as np  
from scipy.optimize import linprog  
from sklearn.metrics.pairwise import cosine\_similarity  
  
# 处理文件夹中的所有文件  
def process\_folder(folder\_path):  
 combined\_array = [] # 初始化为 None  
  
 for filename in os.listdir(folder\_path):  
 if filename.endswith('.xlsx'):  
 filepath = os.path.join(folder\_path, filename)  
 data = pd.read\_excel(filepath)  
  
 array1=extract\_features\_from\_file(data) #求特征1×6的array数组  
 combined\_array.append(array1)  
  
 #assert len(combined\_array) == 60, "数据列表应包含60个1×6的数组"  
  
 # 将数据列表转换为60×6的二维数据矩阵  
 data\_matrix = np.array(combined\_array)  
  
 # 计算相似度矩阵  
 similarity\_matrix = cosine\_similarity(data\_matrix)  
  
 #print(combined\_array)  
 return similarity\_matrix #最后返回的是 二维数组 60×60的 然后进行我们的相关系数计算  
  
  
  
def detect\_outliers(series, alpha=0.05):  
 # 计算平均值和标准差  
 x\_mean = series.mean()  
 x\_std = series.std()  
 Gi = np.abs((series - x\_mean) / x\_std)  
  
 # 查找格拉布斯表获得临界值  
 n = len(series)  
 t\_critical = t.ppf(1 - alpha / (2 \* n), n - 2)  
 G\_critical = ((n - 1) / np.sqrt(n)) \* np.sqrt(t\_critical \*\* 2 / (n - 2 + t\_critical \*\* 2))  
  
 # 剔除大于临界值的异常数据  
 return series[Gi < G\_critical]  
  
  
# 提取特征  
def extract\_features\_from\_file(data):  
  
 # 异常值检测和剔除  
 data['acc\_x(g)'] = detect\_outliers(data['acc\_x(g)'])  
 data['acc\_y(g)'] = detect\_outliers(data['acc\_y(g)'])  
 data['acc\_z(g)'] = detect\_outliers(data['acc\_z(g)'])  
 data['gyro\_x(dps)'] = detect\_outliers(data['gyro\_x(dps)'])  
 data['gyro\_y(dps)'] = detect\_outliers(data['gyro\_y(dps)'])  
 data['gyro\_z(dps)'] = detect\_outliers(data['gyro\_z(dps)'])  
  
 acc\_x = data['acc\_x(g)']  
 acc\_y = data['acc\_y(g)']  
 acc\_z = data['acc\_z(g)']  
 gyro\_x = data['gyro\_x(dps)']  
 gyro\_y = data['gyro\_y(dps)']  
 gyro\_z = data['gyro\_z(dps)']  
  
 #取平均 变成1×6数据  
 acc\_x = acc\_x.mean()  
 acc\_y = acc\_y.mean()  
 acc\_z = acc\_z.mean()  
 gyro\_x = gyro\_x .mean()  
 gyro\_y = gyro\_y.mean()  
 gyro\_z = gyro\_z.mean()  
  
 features = np.array([acc\_x,acc\_y,acc\_z,gyro\_x,gyro\_y,gyro\_z])  
  
 return features #返回我们的数组  
  
  
  
def greedy\_grouping(correlation\_matrix):  
 n = correlation\_matrix.shape[0]  
 grouped = [False] \* n  
 groups = []  
  
 while not all(grouped):  
 group = []  
 # 找到第一个未分组的数据  
 for i in range(n):  
 if not grouped[i]:  
 group.append(i)  
 grouped[i] = True  
 break  
  
 # 选择另外4个数据  
 for \_ in range(4):  
 max\_sum = -np.inf  
 best\_candidate = -1  
 for j in range(n):  
 if not grouped[j]:  
 current\_sum = sum(correlation\_matrix[group[k], j] for k in range(len(group)))  
 if current\_sum > max\_sum:  
 max\_sum = current\_sum  
 best\_candidate = j  
  
 group.append(best\_candidate)  
 grouped[best\_candidate] = True  
  
 groups.append(group)  
  
 return groups  
  
  
  
  
# 主函数  
def main():  
 # 处理每个人的数据  
 acc\_features1 = process\_folder('Person1')  
 acc\_features2 = process\_folder('Person2')  
 acc\_features3 = process\_folder('Person3')  
 # 返回60×60数组  
 print(acc\_features1)  
  
 groups1 = greedy\_grouping(acc\_features1)  
 groups2 = greedy\_grouping(acc\_features2)  
 groups3 = greedy\_grouping(acc\_features3)  
  
 # 将分组结果存储到Excel文件  
 df = pd.DataFrame({f"Group {i + 1}": group for i, group in enumerate(groups1)})  
 df.to\_excel("group\_results1.xlsx", index=False)  
 df = pd.DataFrame({f"Group {i + 1}": group for i, group in enumerate(groups2)})  
 df.to\_excel("group\_results2.xlsx", index=False)  
 df = pd.DataFrame({f"Group {i + 1}": group for i, group in enumerate(groups3)})  
 df.to\_excel("group\_results3.xlsx", index=False)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

格式太难改了，表格序号为0 - 59，文件上序号为SY1 - SY60。

Person1

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Group 1** | **Group 2** | **Group 3** | **Group 4** | **Group 5** | **Group 6** | **Group 7** | **Group 8** | **Group 9** | **Group 10** | **Group 11** | **Group 12** |
| 0 | 1 | 2 | 3 | 4 | 6 | 8 | 9 | 10 | 13 | 14 | 32 |
| 15 | 12 | 22 | 44 | 53 | 26 | 35 | 33 | 38 | 17 | 28 | 52 |
| 57 | 20 | 55 | 54 | 11 | 46 | 19 | 23 | 37 | 48 | 27 | 59 |
| 36 | 41 | 39 | 58 | 29 | 49 | 30 | 50 | 18 | 42 | 34 | 43 |
| 31 | 5 | 7 | 51 | 16 | 45 | 24 | 56 | 40 | 21 | 25 | 47 |

Person2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Group 1** | **Group 2** | **Group 3** | **Group 4** | **Group 5** | **Group 6** | **Group 7** | **Group 8** | **Group 9** | **Group 10** | **Group 11** | **Group 12** |
| 0 | 1 | 2 | 3 | 4 | 5 | 7 | 8 | 10 | 12 | 16 | 24 |
| 43 | 41 | 44 | 28 | 27 | 9 | 37 | 48 | 15 | 31 | 57 | 25 |
| 33 | 18 | 49 | 55 | 42 | 58 | 13 | 50 | 30 | 22 | 52 | 32 |
| 45 | 11 | 14 | 6 | 20 | 35 | 21 | 59 | 23 | 46 | 26 | 54 |
| 38 | 53 | 51 | 36 | 19 | 39 | 34 | 17 | 40 | 56 | 29 | 47 |

Person3

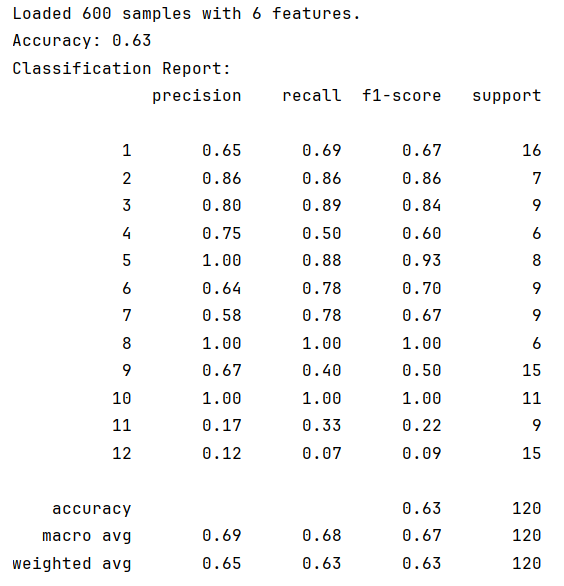
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Group 1** | **Group 2** | **Group 3** | **Group 4** | **Group 5** | **Group 6** | **Group 7** | **Group 8** | **Group 9** | **Group 10** | **Group 11** | **Group 12** |
| 0 | 1 | 2 | 5 | 6 | 8 | 9 | 10 | 14 | 17 | 22 | 27 |
| 34 | 4 | 56 | 37 | 12 | 13 | 52 | 29 | 36 | 24 | 49 | 48 |
| 57 | 42 | 38 | 59 | 18 | 55 | 53 | 28 | 47 | 21 | 50 | 44 |
| 39 | 51 | 3 | 30 | 20 | 26 | 33 | 16 | 32 | 46 | 45 | 35 |
| 40 | 11 | 19 | 23 | 7 | 31 | 54 | 15 | 58 | 41 | 25 | 43 |

# 问题二

这个代码对附件二结果进行预测并存在表格文件中。

import os  
import matplotlib.pyplot as plt  
import pandas as pd  
import pulp  
from scipy.stats import t  
import numpy as np  
from scipy.optimize import linprog  
from sklearn.metrics.pairwise import cosine\_similarity  
#  
#  
#  
def load\_data(folder\_path):  
 data = []  
 labels = []  
 persons = os.listdir(folder\_path)  
 for person in persons:  
 person\_path = os.path.join(folder\_path, person)  
 if os.path.isdir(person\_path):  
 activities = os.listdir(person\_path)  
 for activity in activities:  
 activity\_path = os.path.join(person\_path, activity)  
 if os.path.isfile(activity\_path) and activity.endswith('.xlsx'):  
 df = pd.read\_excel(activity\_path)  
  
 if df.shape[1] >= 6: # 需要至少6列数据  
 features = df.iloc[:, :6].values  
 data.append(features)  
 label = activity.split('t')[0]  
 label = label.split('a')[1]  
 # 提取活动类型  
 #print(label)  
 labels.append(label)  
 else:  
 print(f"Warning: File {activity\_path} does not have enough columns.")  
 return data,label  
  
#  
folder\_path = '附件2'  
data, labels = load\_data(folder\_path)  
  
print(f"Loaded {len(data)} data segments with {len(labels)} labels.")  
import pandas as pd  
from sklearn.metrics import accuracy\_score  
  
  
  
file\_path = 'group\_results4.xlsx'  
df = pd.read\_excel(file\_path)  
  
# 打印数据框的前几行以检查数据  
print(df.head())  
  
  
  
# 计算每组的准确率  
accuracies = {}  
for group in df.columns:  
 pred\_labels = df[group].values  
 # 确保预测标签长度与实际标签长度相匹配  
 if len(pred\_labels) >= len(labels):  
 pred\_labels = pred\_labels[:len(labels)]  
 accuracy = accuracy\_score(labels, pred\_labels)  
 accuracies[group] = accuracy  
 else:  
 print(f"Warning: Length of predictions for {group} is shorter than true labels.")  
  
# 打印每组的准确率  
for group, accuracy in accuracies.items():  
 print(f"{group} Accuracy: {accuracy:.2f}")

这个代码通过随机森林，得到预测模型：



import os  
import pandas as pd  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score, classification\_report  
import numpy as np  
  
  
# 数据加载函数  
def load\_data(folder\_path):  
 data = []  
 labels = []  
 persons = os.listdir(folder\_path)  
 for person in persons:  
 person\_path = os.path.join(folder\_path, person)  
 if os.path.isdir(person\_path):  
 activities = os.listdir(person\_path)  
 for activity in activities:  
 activity\_path = os.path.join(person\_path, activity)  
 if os.path.isfile(activity\_path) and activity.endswith('.xlsx'):  
 df = pd.read\_excel(activity\_path)  
  
 if df.shape[1] >= 6: # 需要至少6列数据  
 features = df.iloc[:, :6].values  
 # 对每个文件提取特征的平均值作为最终特征  
 mean\_features = np.mean(features, axis=0)  
 data.append(mean\_features)  
  
 a=activity.split('t')[0] # 提取活动类型并转换为整数  
 a=a.split('a')[1]  
 label = int(a)  
 labels.append(label)  
 else:  
 print(f"Warning: File {activity\_path} does not have enough columns.")  
 return np.array(data), np.array(labels)  
  
  
# 加载数据  
folder\_path = '附件2'  
data, labels = load\_data(folder\_path)  
print(f"Loaded {data.shape[0]} samples with {data.shape[1]} features.")  
  
# 切分数据集  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(data, labels, test\_size=0.2, random\_state=42)  
  
# 初始化和训练随机森林模型  
clf = RandomForestClassifier(n\_estimators=100, random\_state=42)  
clf.fit(X\_train, y\_train)  
  
# 进行预测  
y\_pred = clf.predict(X\_test)  
  
# 评估模型  
accuracy = accuracy\_score(y\_test, y\_pred)  
report = classification\_report(y\_test, y\_pred)  
  
print(f"Accuracy: {accuracy:.2f}")  
print("Classification Report:")  
print(report)

这个代码通过我们训练的模型将预测的附件三分类存到表格文件中

import os  
import pandas as pd  
import numpy as np  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.preprocessing import StandardScaler  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import classification\_report  
  
  
# 数据加载函数  
def load\_data(folder\_path):  
 data = []  
 labels = []  
 persons = os.listdir(folder\_path)  
 for person in persons:  
 person\_path = os.path.join(folder\_path, person)  
 if os.path.isdir(person\_path):  
 activities = os.listdir(person\_path)  
 for activity in activities:  
 activity\_path = os.path.join(person\_path, activity)  
 if os.path.isfile(activity\_path) and activity.endswith('.xlsx'):  
 df = pd.read\_excel(activity\_path)  
  
 if df.shape[1] >= 6: # 需要至少6列数据  
 features = df.iloc[:, :6].values  
 mean\_features = np.mean(features, axis=0)  
 data.append(mean\_features)  
 a=activity.split('t')[0] # 提取活动类型并转换为整数  
 a=a.split('a')[1]  
 label = int(a)  
 labels.append(label)  
 else:  
 print(f"Warning: File {activity\_path} does not have enough columns.")  
 return np.array(data), np.array(labels)  
  
  
# 训练随机森林模型  
def train\_random\_forest(X\_train, y\_train):  
 clf = RandomForestClassifier(n\_estimators=100, random\_state=42)  
 clf.fit(X\_train, y\_train)  
 return clf  
  
  
# 对附件三进行分类  
def classify\_new\_data(model, new\_data\_path):  
 new\_data = []  
 files = os.listdir(new\_data\_path)  
 for file in files:  
 file\_path = os.path.join(new\_data\_path, file)  
 if os.path.isfile(file\_path) and file.endswith('.xlsx'):  
 df = pd.read\_excel(file\_path)  
 if df.shape[1] >= 6: # 需要至少6列数据  
 features = df.iloc[:, :6].values  
 mean\_features = np.mean(features, axis=0)  
 new\_data.append(mean\_features)  
 else:  
 print(f"Warning: File {file\_path} does not have enough columns.")  
 new\_data = np.array(new\_data)  
  
 # 标准化数据  
 scaler = StandardScaler()  
 new\_data\_scaled = scaler.fit\_transform(new\_data)  
  
 # 进行预测  
 predictions = model.predict(new\_data\_scaled)  
 return predictions  
  
  
# 加载附件二的数据用于训练  
folder\_path = '附件2'  
data, labels = load\_data(folder\_path)  
  
# 切分数据集  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(data, labels, test\_size=0.2, random\_state=42)  
  
# 训练模型  
model = train\_random\_forest(X\_train, y\_train)  
  
# 加载附件三的数据进行分类  
new\_data\_path = '附件3'  
predictions = classify\_new\_data(model, new\_data\_path)  
  
# 假设附件三中的文件名可以作为活动类型的标识  
activity\_types = [f'SY{i + 1}' for i in range(len(predictions))]  
  
# 创建 DataFrame 以保存结果  
results\_df = pd.DataFrame({  
 'Activity Type': activity\_types,  
 'Predicted Status': predictions  
})  
  
# 保存结果到 Excel  
results\_df.to\_excel('predictions\_for\_attachment3.xlsx', index=False)  
  
# 打印预测报告  
print("Predictions have been saved to 'predictions\_for\_attachment3.xlsx'.")

# 问题三

import pandas as pd  
import numpy as np  
from scipy.stats import f\_oneway  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.preprocessing import LabelEncoder  
import os  
  
  
# 1. 加载实验人员数据  
def load\_person\_data(person\_file):  
 person\_df = pd.read\_excel(person\_file)  
 # 处理身高和体重  
 person\_df['身高'] = person\_df['身高'].str.replace('kg', '', regex=False).astype(float)  
 person\_df['体重'] = person\_df['体重'].str.replace('cm', '', regex=False).astype(float)  
 return person\_df  
  
  
# 2. 加载活动数据  
def load\_activity\_data(activity\_folder):  
 data = []  
 labels = []  
 persons = os.listdir(activity\_folder)  
 for person in persons:  
 person\_path = os.path.join(activity\_folder, person)  
 if os.path.isdir(person\_path):  
 activities = os.listdir(person\_path)  
 for activity in activities:  
 activity\_path = os.path.join(person\_path, activity)  
 if os.path.isfile(activity\_path) and activity.endswith('.xlsx'):  
 df = pd.read\_excel(activity\_path)  
 if df.shape[1] >= 6:  
 features = df.iloc[:, :6].values  
 mean\_features = np.mean(features, axis=0)  
 data.append(mean\_features)  
 a=activity.split('t')[0] # 提取活动类型并转换为整数  
 a=a.split('a')[1]  
 label = int(a)  
 labels.append(label)  
 return np.array(data), np.array(labels)  
  
  
# 3. 分析不同人员的同一活动状态是否存在差异  
def analyze\_activity\_differences(person\_file, activity\_folder):  
 person\_df = load\_person\_data(person\_file)  
 data, labels = load\_activity\_data(activity\_folder)  
  
 activity\_person\_df = pd.DataFrame(data,  
 columns=['Feature1', 'Feature2', 'Feature3', 'Feature4', 'Feature5', 'Feature6'])  
 activity\_person\_df['Label'] = labels  
 activity\_person\_df = activity\_person\_df.merge(person\_df, left\_on='Label', right\_on='实验人员编号')  
  
 # 根据实验人员编号分组  
 grouped = activity\_person\_df.groupby('实验人员编号')  
 for person\_id, group in grouped:  
 print(f"Person ID: {person\_id}")  
 print(group[['年龄', '身高', '体重']].describe())  
  
 # 使用 ANOVA 测试分析年龄、身高、体重的差异  
 age\_groups = [group['年龄'] for person\_id, group in grouped]  
 f\_stat, p\_val = f\_oneway(\*age\_groups)  
 print(f"ANOVA Age Test: F-statistic = {f\_stat}, p-value = {p\_val}")  
  
  
# 4. 分析活动状态数据与实验人员的个人信息关系  
def analyze\_correlations(person\_file, activity\_folder):  
 person\_df = load\_person\_data(person\_file)  
 data, labels = load\_activity\_data(activity\_folder)  
  
 activity\_person\_df = pd.DataFrame(data,  
 columns=['Feature1', 'Feature2', 'Feature3', 'Feature4', 'Feature5', 'Feature6'])  
 activity\_person\_df['Label'] = labels  
 activity\_person\_df = activity\_person\_df.merge(person\_df, left\_on='Label', right\_on='实验人员编号')  
  
 # 计算相关性  
 correlation = activity\_person\_df[  
 ['年龄', '身高', '体重', 'Feature1', 'Feature2', 'Feature3', 'Feature4', 'Feature5', 'Feature6']].corr()  
 print("Correlation Matrix:")  
 print(correlation)  
  
  
# 5. 使用随机森林模型进行人员画像  
def train\_random\_forest(X\_train, y\_train):  
 clf = RandomForestClassifier(n\_estimators=100, random\_state=42)  
 clf.fit(X\_train, y\_train)  
 return clf  
  
  
def predict\_person(clf, X\_test):  
 return clf.predict(X\_test)  
  
  
# 6. 使用模型对文件夹中的数据进行分类  
def classify\_folder\_data(model, folder\_path):  
 predictions = {}  
 files = os.listdir(folder\_path)  
 for file\_name in files:  
 if file\_name.endswith('.xlsx'):  
 file\_path = os.path.join(folder\_path, file\_name)  
 new\_data = pd.read\_excel(file\_path)  
 X\_new = new\_data.values  
 pred = model.predict(X\_new)  
 predictions[file\_name] = pred  
 return predictions  
  
  
# 主程序  
if \_\_name\_\_ == "\_\_main\_\_":  
 person\_file = '附件4.xlsx'  
 activity\_folder = '附件2'  
 new\_data\_folder = '附件5'  
  
 # 分析人员差异  
 analyze\_activity\_differences(person\_file, activity\_folder)  
  
 # 分析活动状态与个人信息的关系  
 analyze\_correlations(person\_file, activity\_folder)  
  
 # 准备训练数据和模型  
 X\_train, y\_train = load\_activity\_data(activity\_folder)  
 clf = train\_random\_forest(X\_train, y\_train)  
  
 # 对附件 5 中的数据进行分类  
 predictions = classify\_folder\_data(clf, new\_data\_folder)  
  
 # 将结果填入表 3  
 result\_df = pd.DataFrame({  
 '活动类型': list(predictions.keys()),  
 '判别结果': [list(pred)[0] for pred in predictions.values()]  
 })  
 result\_df.to\_excel('表3.xlsx', index=False)  
 print("Results saved to 表3.xlsx")