机器学习实验报告

姓名：杨崇焕

学号：U201610531

班级：电信中英1601

实验课题：决策树

一.实验目的：

```
使用决策树来完成如下三个分类预测问题。
   任务一：使用决策树预测隐形眼镜类型
   任务二：根据用户采集的WiFi信息采用决策树预测用户所在房间
   任务三：IMDB数据集电影评测分类（二分类问题）
```

二.实验原理：

```
决策树的组成   ：决策结点(样本属性，非叶子结点)，叶子结点(类别标签)，属性分枝(属性测试)
决策(分类过程)：从根结点出发，沿着决策树从上到下的遍历；每个非叶子结点相当
               一个if条件语句，因此在每个非叶子结点都有一个属性测试。测试结
               果对应不同的分枝，最后抵达一个叶子结点，从而得到分类结果。
决策树类型     :  按判决条件分为 ID3决策树，C45决策树，CART决策树
决策树的优化   :  预剪枝，后剪枝。
```

三.实验过程：

实验环境：

- ubuntu 18.04
- python 3.6
- numpy 1.14.3
- pandas 0.23.0
- scikit-learn 0.19.1
- pydotplus
- graphviz

任务一：使用决策树预测隐形眼镜类型

隐形眼镜数据集是非常著名的数据集，它包含了很多患者眼部状况的观察条件以及医生推荐的隐形眼镜类型。隐形眼镜类型包括硬材质（hard）、软材质（soft）以及不适合佩戴隐形眼镜（no lenses）。以下为该数据集的部分数据，包括年龄、近视or远视类型，是否散光，是否容易流泪，最后1列为应佩戴眼镜类型。使用决策树来对佩戴隐形眼镜类型进行预测。

导入模板库：

```
    from sklearn import tree
    from sklearn.preprocessing import OneHotEncoder,LabelEncoder
    from sklearn.externals.six import StringIO
    import pandas as pd
    import pydotplus
```

加载处理数据：

```
用Python解析文本文件，解析tab键分割的数据行，用dict储存类别及标签，转换成pandas.DataFrame，再利用one-hot编码
```

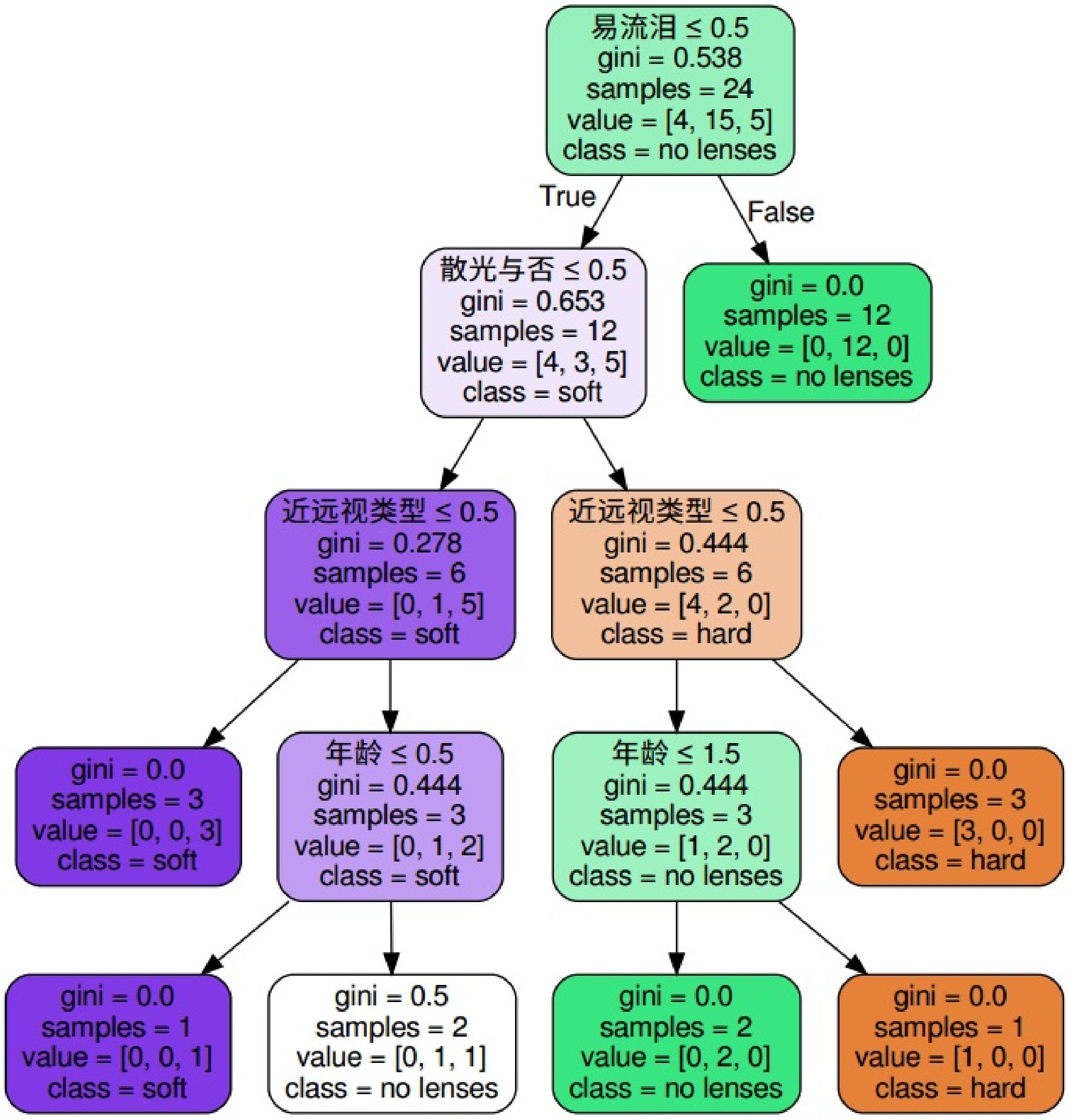
```
#提取出target label
with open('lenses.txt','r') as fr:
    lenses = [ line.strip().split('\t') for line in fr.readlines()]
lenses_target = []
for row in lenses:
    lenses_target.append(row[-1])
#设置类别标签
lenses_Label = ['age', 'prescript', 'astigmatic', 'tearRate']
lenses_list = []
lenses_dict = {}
# get the result for each label then store them in a dict --> lenses_dict
#获取每个类别标签对应的结果，并保存在对应的字典项的值中
for each_label in lenses_Label:
    for row in lenses:
        lenses_list.append( row[lenses_Label.index(each_label)] )
    lenses_dict[each_label] = lenses_list
    lenses_list = []
#将结果转化成pandas.DataFrame ,并用one-hot encoder进行处理编码
lenses_pd = pd.DataFrame(lenses_dict)

le = LabelEncoder()
for column in lenses_pd.columns:
    lenses_pd[column] = le.fit_transform(lenses_pd[column])
```

训练决策树：

```
clf = tree.DecisionTreeClassifier(max_depth=4)
clf = clf.fit(lenses_pd.values.tolist(),lenses_target)
```

利用graphviz打印决策树

```
dot_data = StringIO()
tree.export_graphviz(clf, out_file = dot_data,
                    feature_names = lenses_pd.keys(),
                    class_names = clf.classes_,
                    filled=True, rounded=True,
                    special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_pdf("tree2.pdf")
```



任务二：根据用户采集的WiFi信息采用决策树预测用户所在房间

数据处理：

将训练集导入为pandas.DataFrame， 并用BSSID_type, Room_type,fin_type,分别记录
BSSIDLabel,RoomLabel,finLabel
的信息，此处处理数据时将不明显特征SSIDLabel舍弃，分布过于集中的RSSLabel舍弃，仅考虑在某时某地接收到BSSID的种类，
利用one-hot encode在训练集空间中编码，出现的记为１，未出现的记为０。同时在测试时样本空间中有而训练集空间中没有的
BSSID舍去不计。

```
#训练导入数据
TrainSet = pd.read_csv('TrainDT.csv',encoding='gbk')
fin_type = TrainSet.iloc[:,-1].unique()
BSSID_type = TrainSet.iloc[:,0].unique()
Room_type = TrainSet.iloc[:,2].unique()
```

在训练集空间对BSSID进行one-hot encode

```
fin = {}
BSSID_list = np.asarray([0 for i in range(len(BSSID_type))])
for fin_i in fin_type:
    indexx = []
    TD_index = pd.Index(TrainSet.iloc[:,-1])
    sliced_TD = TrainSet[TD_index.get_loc(fin_i)]
    for i in sliced_TD.iloc[:,0]:
        index = pd.Index(BSSID_type).get_loc(i)
        #indexx.extend([index])
        indexx.append(index)
    BSSID_list[np.asarray(indexx)] = 1
    fin[fin_i] = []
    fin[fin_i].extend(BSSID_list.tolist())
    # add label at the end
    fin[fin_i].extend([sliced_TD.iloc[0,2]])
```

编码结果保存为pandas DataFrame

```
Label_name = []
BSSID_name = BSSID_type.tolist()
Label_name.extend(BSSID_name)
BSSID_name.extend(['target'])

pdf_fin = pd.DataFrame(fin).T
pdf_fin.columns =  BSSID_name
feature_data = pdf_fin.iloc[:,0:len(BSSID_type)]
target = pdf_fin.iloc[:,len(BSSID_type)]
```

导入测试集

```
TestSet = pd.read_csv('TestDT.csv',encoding='gbk')
t_fin_type = TestSet.iloc[:,-1].unique()
t_BSSID_type = TestSet.iloc[:,0].unique()
t_Room_type = TestSet.iloc[:,2].unique()
```

在样本空间进行one-hot encode

```
t_fin = {}
t_BSSID_list = np.asarray([0 for i in range(len(BSSID_type))])
for fin_i in t_fin_type:
    indexx = []
    TD_index = pd.Index(TestSet.iloc[:,-1])
    sliced_TD = TestSet[TD_index.get_loc(fin_i)]
    for i in sliced_TD.iloc[:,0]:
        if i in BSSID_type:
            index = pd.Index(BSSID_type).get_loc(i)
            indexx.append(index)
    t_BSSID_list[np.asarray(indexx)] = 1
    t_fin[fin_i] = []
    t_fin[fin_i].extend(t_BSSID_list.tolist())
    t_fin[fin_i].extend([sliced_TD.iloc[0,2]])
```

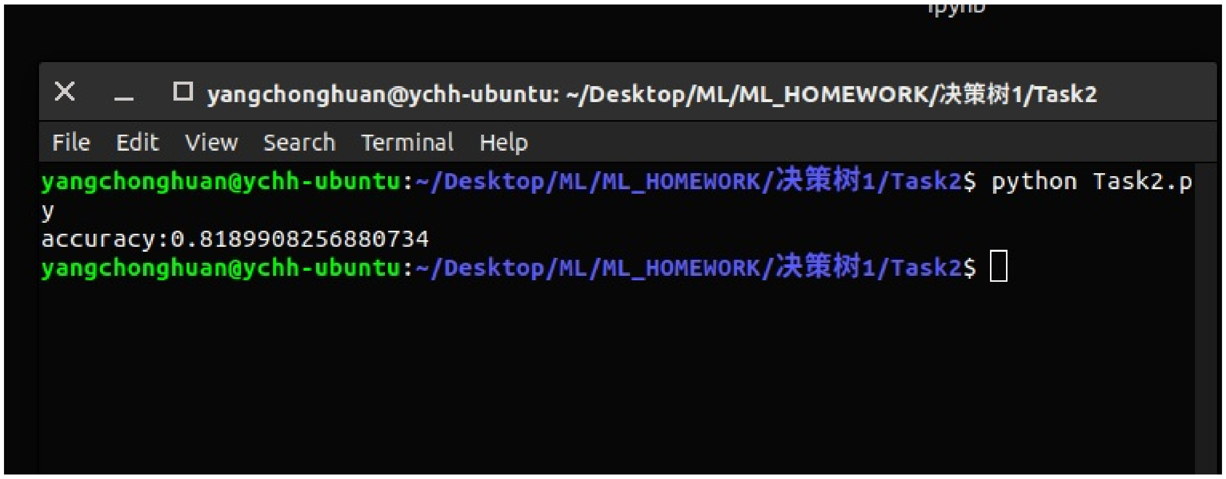
将测试集处理结果保存为pandas DataFrame

```
t_Label_name = []
t_BSSID_name = BSSID_type.tolist()
t_Label_name.extend(BSSID_name)
t_BSSID_name.extend(['target'])

t_pdf_fin = pd.DataFrame(t_fin).T
t_pdf_fin.columns =  t_BSSID_name
t_feature_data = t_pdf_fin.iloc[:,0:len(BSSID_type)]
t_target = t_pdf_fin.iloc[:,len(BSSID_type)]
```

预测结果与测试集结果对比得到精度，同时循环训练测试结果１００，取平均精度

```
for i in range(100):
    clf = tree.DecisionTreeClassifier(max_depth = None)
    clf = clf.fit(feature_data,target)
    pred_target = []
    for i in range(t_feature_data.index.size):
        m = clf.predict([t_feature_data.iloc[i,:]]).tolist()
        pred_target.extend(m)
    summ = 0
    target_list = t_target.tolist()
    for i in range(t_feature_data.index.size):
        if pred_target[i] == target_list[i]:
            summ += 1
    accuracy += summ / t_feature_data.index.size
print("accuracy:{}".format(accuracy/100))
```

得到结果



```
yangchonghuan@ychh-ubuntu: ~/Desktop/ML/ML_HOMEWORK/决策树1/Task2
File  Edit  View  Search  Terminal  Help
yangchonghuan@ychh-ubuntu:~/Desktop/ML/ML_HOMEWORK/决策树1/Task2$ python Task2.p
y
accuracy:0.8189908256880734
yangchonghuan@ychh-ubuntu:~/Desktop/ML/ML_HOMEWORK/决策树1/Task2$ 
```

任务二：IMDB数据集电影评测分类（二分类问题）

数据处理：

```
用one-hot编码，设词库大小为10000，因此转换的one-hot编码也是10000维，出现的设为１，未出现的设为０.
将训练集拆为4/5的训练集，与１/5的测试集，用测试集上最优树来进行分类
```

导入train_data, train_label

```
train_data = pd.read_csv('./train/train_data.txt',sep='\t',header=None)
train_label = pd.read_csv('./train/train_labels.txt',sep='\t',header=None)
```

利用one-hot encode来对每个评论编码

```
with open('./train/train_data.txt','r') as fr:
    train_data_list = [line.strip().split('\t') for line in fr.readlines()]

vector = [0 for i in range(10000)]
train_data_dict = {}
for i in range(train_data.index.size):
    for item in [int(s) for s in train_data_list[i][0].split() if s.isdigit()]:
        vector[item] = 1
    train_data_dict[i]=[]
    train_data_dict[i].extend(vector)
    vector = [0 for i in range(10000)]
```

拆分训练集

```
# split the data set to training set and testing set
TrainSet = pd.DataFrame(train_data_dict).T.iloc[0:20000,:]
train_target = train_label.iloc[0:20000,:]
TestSet = pd.DataFrame(train_data_dict).T.iloc[20000:25000,:]
target = train_label.iloc[20000:25000,:]
```

寻找最佳树深度来的测试集最优结果

```
    acc = []
    for depth in range(4,100):
        clf = tree.DecisionTreeClassifier(max_depth= depth)
        clf = clf.fit(TrainSet,train_target)
        pred_target = []
        for i in range(5000):
            m = clf.predict([TestSet.iloc[i,:]]).tolist()
            pred_target.extend(m)
        summ = 0
        target_list = target.iloc[:,0].tolist()
        for i in range(5000):
            if pred_target[i] == target_list[i]:
                summ += 1
        accuracy = summ / 5000
        print("depth = {}\n".format(depth))
        print("accuracy = {}\n".format(accuracy))
        acc.extend([accuracy])
```

训练

```
m_index = acc.index(max(acc))
clf = tree.DecisionTreeClassifier(max_depth = m_index+4)
```

获取test data

```
    # process the test set
    true_test_data = pd.read_csv('./test/test_data.txt',sep='\t',header=None)
    with open('./test/test_data.txt','r') as fr:
        test_data_list = [line.strip().split('\t') for line in fr.readlines()]
    test_data_dict = {}
```

利用one-hot encode编码

```
    for i in range(true_test_data.index.size):
        for item in [int(s) for s in test_data_list[i][0].split() if s.isdigit()]:
            vector[item] = 1
        test_data_dict[i]=[]
        test_data_dict[i].extend(vector)
        vector = [0 for i in range(10000)]
    true_TestSet = pd.DataFrame(train_data_dict)
    pred_target = []
```

预测并储存结果

```
    for i in range(true_test_data.index.size):
        m = clf.predict([true_TestSet.iloc[:,i]]).tolist()
        pred_target.extend(m)
    np.savetxt("test.txt", pred_target,fmt = '%d')
    print('done!')
```

四.实验小结：

通过本次实验中，我在使用sklearn之前实现过决策树，在这个过程中我反复琢磨决策树的实现细节加深了对决策树的理解。同时通过这三个任务，我学习了利用机器学习处理问题的大致过程，提高了我的动手能力。美中不足的是在这个过程中鉴于时间问题我并没有实现决策树的剪枝优化部分。