大数据导论实验

实验二
数据理解、数据预处理及决策树的应用

主讲教师：叶允明

实验教师：谢佳、房敏

本学期实验总体安排

实验课程共4个学时，2个实验项目，总成绩为30分。

实验一（15分）

Hadoop环境配置与MapReduce编程

掌握Hadoop分布式环境的配置方法；理解Mapreduce作业的原理和编程方法。

实验二（15分）

数据理解、数据预处理及决策树的应用

通过应用案例实践数据预处理方法；编码实现一个经典的数据挖掘算法。

目录

实验二任务

实验任务：个人年收入预测

Adult Data Set数据集是Barry Becker从1994年的人口普查数据库中整理筛选得到的，数据收集和整理过程规范统一，样本量充足且缺失值、异常值较少，能够一定程度上保证模型的质量和可信度。

https://archive.ics.uci.edu/ml/datasets/Adult

数据集	人口普查收入
记录数量	48842
属性	14
类别	2

该数据集对应的任务是二分类任务，分类目标是预测一个人年收入(income)是否超过50k美元 。

adult.data

32561条

adult.test

16281条

数据说明

属性	描述	类型
age	年龄	连续型
workclass	工作类别	离散型
fnlwgt	样本权重	连续型
education	受教育程度	离散型
education-num	受教育时间	连续型
marital-status	婚姻状况	离散型
occupation	职业	离散型

数据说明

属性	描述	类型
relationship	社会角色	离散型
race	种族	离散型
sex	性别	离散型
capital-gain	资本收益	连续型
capital-loss	资本支出	连续型
hours-per-week	每周工作时间	连续型
native-country	国籍	离散型
income（目标）	年收入	离散型

预备知识

Pandas库的介绍

Pandas是python第三方库，它是一个强大的分析结构化数据的工具集；它的使用基础是Numpy（提供高性能的矩阵运算），用于数据挖掘和数据分析，同时也提供数据清洗功能。

```
pip install pandas
```

工欲善其事必先利其器！

预备知识

Pandas的核心数据结构

维数	名称	描述
1	Series	带标签的一维同构数组
2	DataFrame	带标签的，大小可变的，二维异构表格

- Series 是带标签的一维数组，可存储整数、浮点数、字符串、Python 对象等类型的数据。
- DataFrame 是一个表格型的数据结构，类似于 Excel 或 sql 表，它含有一组有序的列，每列可以是不同的值类型（数值、字符串、布尔值等）。

预备知识

Pandas库的基本操作

- 查看数据
  df.describe()、df.index 、df.columns 、df.values
- 选取特定列和行的数据
  df[col]、df[[col1,col2]]、df.iloc[[row1, row2], [col1, col2]]
- 增加、删除、修改列的值
  del df[col] 、df.pop
- 导入导出文件
  pd.read_csv、df.to_csv

更多学习请参考pandas中文网： https://www.pypandas.cn/

实验步骤

数据读取

① 将数据处理成csv格式，并保存

```
columns = ['age', 'workclass', 'fnlwgt', 'education', 'educationNum', 'maritalStatus', 'occupation', 'relationship', 'ra
           'capitalGain', 'capitalLoss', 'hoursPerWeek', 'nativeCountry', 'income']
df_train_set = pd.read_csv('./adult.data', names=columns)
df_test_set = pd.read_csv('./adult.test', names=columns, skiprows=1) #第一行是非法数据

print(df_train_set.head())
#print(df_test_set.head())
df_train_set.to_csv('./train_adult.csv', index=False)
df_test_set.to_csv('./test_adult.csv', index=False)
```

```
   age          workclass  fnlwgt   education  educationNum  \
0   39          State-gov   77516   Bachelors            13
1   50   Self-emp-not-inc   83311   Bachelors            13
2   38            Private  215646     HS-grad             9
3   53            Private  234721        11th             7
4   28            Private  338409   Bachelors            13

        maritalStatus          occupation    relationship    race      sex  \
0       Never-married        Adm-clerical   Not-in-family   White     Male
1   Married-civ-spouse     Exec-managerial         Husband   White     Male
2             Divorced   Handlers-cleaners   Not-in-family   White     Male
3   Married-civ-spouse   Handlers-cleaners         Husband   Black     Male
4   Married-civ-spouse      Prof-specialty            Wife   Black   Female

   capitalGain  capitalLoss  hoursPerWeek   nativeCountry  income
0         2174            0            40   United-States   <=50K
1            0            0            13   United-States   <=50K
2            0            0            40   United-States   <=50K
3            0            0            40   United-States   <=50K
4            0            0            40            Cuba   <=50K
```

实验步骤

1 数据读取

② 获取所有的表头和数据信息，转化为DataFrame格式

```
df_train_set = pd.read_csv('./train_adult.csv')
df_train_set
```

	age	workclass	fnlwgt	education	educationNum	maritalStatus	occupation	relationship	race	sex	capitalGain	capitalLoss	hc
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	

32561 rows × 15 columns

实验步骤

数据读取

③ 查看数据类型

```
df_test_set.dtypes
```

```
age               int64
workclass        object
fnlwgt            int64
education        object
educationNum      int64
maritalStatus    object
occupation       object
relationship     object
race             object
sex              object
capitalGain       int64
capitalLoss       int64
hoursPerWeek      int64
nativeCountry    object
income           object
dtype: object
```

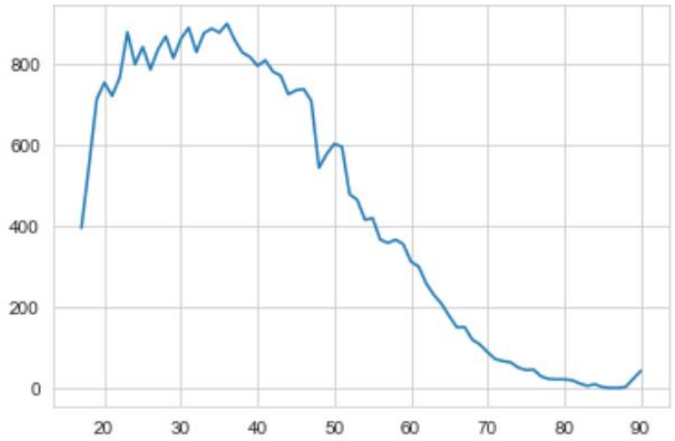
实验步骤

数据读取

④ 数据可视化

```
df_train_set['age'].value_counts().sort_index().plot.line()
```

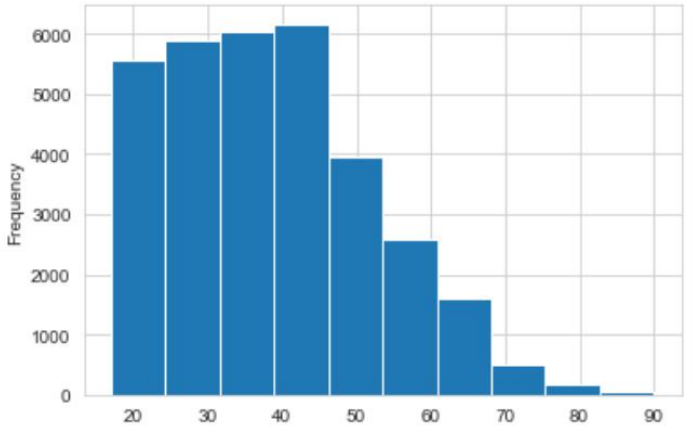
```
<AxesSubplot:>
```



折线图

```
df_train_set['age'].plot.hist()
```

```
<AxesSubplot:ylabel='Frequency'>
```



柱状图

实验步骤

数据预处理

① 删除指定属性

```
# fnlwgt列用处不大，educationNum与education类似
df_train_set.drop(['fnlwgt', 'educationNum'], axis=1, inplace=True)
```

```
df_train_set.columns
```

```
Index(['age', 'workclass', 'education', 'maritalStatus', 'occupation',
       'relationship', 'race', 'sex', 'capitalGain', 'capitalLoss',
       'hoursPerWeek', 'nativeCountry', 'income'],
      dtype='object')
```

② 重复记录处理

```
df_train_set.drop_duplicates(inplace=True)# 去除重复的行
df_train_set.shape[0]#获取行数
```

```
29096
```

③ 缺失值处理（无缺失值）

```
df_train_set.dropna(inplace=True)#删除包含缺失值的行
df_train_set.shape[0] #获取行数
```

```
29096
```

实验步骤

数据预处理

④ 异常值处理

```
df_train_set[df_train_set['workclass'].str.contains(r'\?', regex=True)] #查找workclass为异常值的记录
```

	age	workclass	education	maritalStatus	occupation	relationship	race	sex	capitalGain	capitalLoss	hoursPerWeek	na
27	54	?	Some-college	Married-civ-spouse	?	Husband	Asian-Pac-Islander	Male	0	0	60	
61	32	?	7th-8th	Married-spouse-absent	?	Not-in-family	White	Male	0	0	40	
69	25	?	Some-college	Never-married	?	Own-child	White	Male	0	0	40	U
77	67	?	10th	Married-civ-spouse	?	Husband	White	Male	0	0	2	U
106	17	?	10th	Never-married	?	Own-child	White	Female	34095	0	32	U
...	...	...	...	...	...	...	...	...	...	...	...	
32530	35	?	Bachelors	Married-civ-spouse	?	Wife	White	Female	0	0	55	U
32531	30	?	Bachelo									
32539	71	?	Doctora									
32541	41	?	HS-gra									
32542	72	?	HS-gra									

1632 rows × 13 columns

```
#删除有异常值的行
new_columns = ['workclass', 'education', 'maritalStatus', 'occupation', 'relationship', 'race', 'sex',
               'nativeCountry', 'income']
for col in new_columns:
        df_train_set = df_train_set[~df_train_set[col].str.contains(r'\?', regex=True)]
df_train_set.shape[0]
```

```
26904
```

实验步骤

数据预处理

⑤ 离散型属性处理

```
#查看workclass属性的取值和记录数
df_train_set['workclass'].value_counts()
```

```
 Private             19214
 Self-emp-not-inc     2431
 Local-gov            2014
 State-gov            1253
 Self-emp-inc         1049
 Federal-gov           929
 Without-pay            14
Name: workclass, dtype: int64
```

```
df_train_set['workclass'].head()
```

```
0            State-gov
1     Self-emp-not-inc
2              Private
3              Private
4              Private
Name: workclass, dtype: object
```

```
workclass_mapping = {' Private': 0, ' Self-emp-not-inc': 1, ' Self-emp-inc': 1, ' Local-gov': 2,
                     ' State-gov': 2, ' Federal-gov': 2, ' Without-pay': 3, ' Never-worked': 3}
df_train_set['workclass'] = df_train_set['workclass'].map(workclass_mapping)
```

```
df_train_set['workclass'].head()
```

```
0    2
1    1
2    0
3    0
4    0
Name: workclass, dtype: int64
```

实验步骤

数据预处理

⑥ 连续型属性处理

```
df_train_set['age'].max(), df_train_set['age'].min()
```

```
(90, 17)
```

```
df_train_set['age'].head()
```

```
0    39
1    50
2    38
3    53
4    28
Name: age, dtype: int64
```

```
bins = [0, 25, 50, 75, 100] # 分箱区间左开右闭 (0, 25], (25, 50]
df_train_set['age'] = pd.cut(df_train_set['age'], bins, labels=False)
```

```
df_train_set['age'].head()
```

```
0    1
1    1
2    1
3    2
4    1
Name: age, dtype: int64
```

实验步骤

数据预处理

⑦ 保存数据

```
df_train_set.to_csv('./after_train_adult.csv',index=False)
df_train_set
```

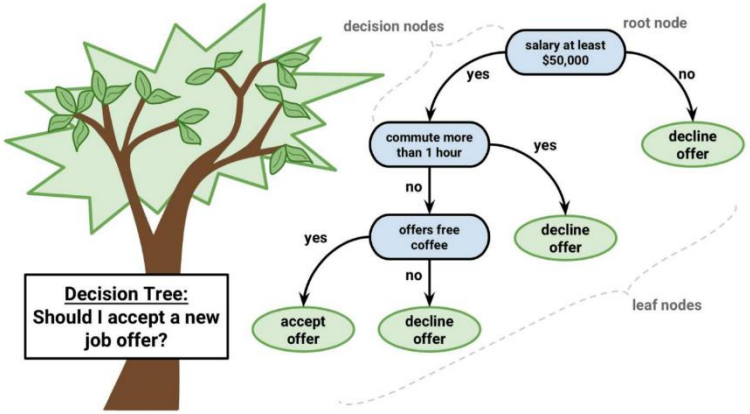
	age	workclass	education	maritalStatus	occupation	relationship	race	sex	capitalGain	capitalLoss	hoursPerWeek	nativeCountry	income
0	1	2	2	1	2	2	0	0	1	0	1	1	0
1	1	1	2	0	2	0	0	0	0	0	0	1	0
2	1	0	1	1	0	2	0	0	0	0	1	1	0
3	2	0	1	0	0	0	1	0	0	0	1	1	0
4	1	0	2	0	2	1	1	1	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
32554	2	0	2	0	2	0	0	0	0	0	1	1	1
32555	0	0	2	1	0	2	0	0	0	0	1	1	0
32556	1	0	1	0	2	1	0	1	0	0	0	1	0
32558	2	0	1	1	2	2	0	1	0	0	1	1	0
32560	2	1	1	0	2	1	0	1	1	0	1	1	1

26904 rows × 13 columns

实验步骤

构建决策树

决策树是一种树形结构的分类器， 树内部的每一个节点代表的是对一个特征的测试，树的分支代表该特征的每一个测试结果，而树的每一个叶子节点代表一个类别。通常根据特征的信息增益或其他指标，构建一棵决策树。



实验步骤

构建决策树

一棵决策树的生成过程主要分为以下3个部分:

（1）特征选择：特征选择是指从训练数据中众多的特征中选择一个特征作为当前节点的分裂标准，如何选择特征有着很多不同量化评估标准（信息增益、信息增益率、基尼系数等），从而衍生出不同的决策树算法。

（2）决策树生成： 根据选择的特征评估标准，从上至下递归地生成子节点，直到数据集不可分则停止决策树停止生长。

（3）剪枝：决策树容易过拟合，一般来需要剪枝，缩小树结构规模、缓解过拟合。剪枝技术有预剪枝和后剪枝两种。

实验步骤

构建决策树

实验任务：

请编码实现某种决策树算法，解决个人年收入的分类问题。

实验要求：

① 编程语言不限
② 决策树算法不限， 可选择ID3、C4.5、C5.0、CART等
③ 不允许调用sklearn等现成库
④ 训练集和测试集可自行设计，采用剪枝、交叉验证等可作为加分项
⑤ 输入训练数据和相关参数构建决策树
⑥ 输入测试数据，使用训练好的决策树进行预测，输出预测结果和准确率

实验步骤

构建决策树

提交要求：

① **实验报告**内容包括预处理的过程及结果、构建决策树的基本过程、测试结果和预测准确率。决策树构建过程需描述训练集和测试集的划分方法，特征选择的方法，剪枝的方法等。

② 复现实验结果所需文件，包含**决策树代码**和相应的**数据文件**。

③ 测试集预测**结果文件**

补充内容

调用sklearn构建决策树：

① 读取数据

```
from sklearn.tree import DecisionTreeClassifier as DTC, export_graphviz
# 读取数据
df = pd.read_csv('D:/大数据/实验二/after_bank.csv')
df = df.iloc[:,1:]
cols = list(df.columns.values)
cols.remove('y')
X = df[cols]
y = df[['y']]
```

② 划分训练集和测试集

```
# 划分训练集与测试集
X_train = X[:4000]
y_train = y[:4000]
X_test = X[4000:5000]
y_test = y[4000:5000]
```

注意：此部分仅作补充内容， 本次实验不允许调用sklearn库！！！

补充内容

③ 训练模型

```
dtc = DTC(criterion='entropy', max_depth=5)  # 基于信息熵
dtc.fit(X_train, y_train)
print('准确率: ', dtc.score(X_test, y_test))
```

```
准确率:  0.8886756238003839
```

④ 参数调整

```
DecisionTreeClassifier(criterion="gini",
                       splitter="best",
                       max_depth=None,
                       min_samples_split=2,
                       min_samples_leaf=1,
                       min_weight_fraction_leaf=0.,
                       max_features=None,
                       random_state=None,
                       max_leaf_nodes=None,
                       min_impurity_decrease=0.,
                       min_impurity_split=None,
                       class_weight=None,
                       presort=False)
```

补充内容

- 通常来说，较为重要的参数有：

criterion：用以设置用信息熵还是基尼系数计算
string, optional (default="gini")
(1)criterion='gini'，分裂节点时评价准则是Gini指数。
(2)criterion='entropy'，分裂节点时的评价指标是信息增益。

splitter：指定分支模式
string, optional (default="best")
(1)splitter='best'，表示选择最优的分裂策略。
(2)splitter='random'，表示选择最好的随机切分策略。

max_depth：指定树的最大深度，防止过拟合
int or None, optional (default=None)
如果为None，表示树的深度不限。直到所有的叶子节点都是纯净的，即叶子节点中所有的样本点都属于同一个类别。或者每个叶子节点包含的样本数小于min_samples_split。

min_samples_leaf：限定每个节点分枝后子节点至少有多少个数据，否则就不分枝
int, float, optional (default=2)
如果为整数，则min_samples_split就是最少样本数。
如果为浮点数(0到1之间)，则每次分裂最少样本数为ceil(min_samples_split * n_samples)。

补充内容

- 不同参数对结果的影响：

```
dtc = DTC(criterion='gini', max_depth=5)  # 基于基尼系数
dtc.fit(X_train, y_train)
print('准确率: ', dtc.score(X_test, y_test))
```

```
准确率:  0.8925143953934741
```

```
for depth in range(1, 10):
    dtc = DTC(criterion='entropy', max_depth=depth)  # 基于信息熵
    dtc.fit(X_train, y_train)
    print('depth:', depth, '|', '准确率:', dtc.score(X_test, y_test))
```

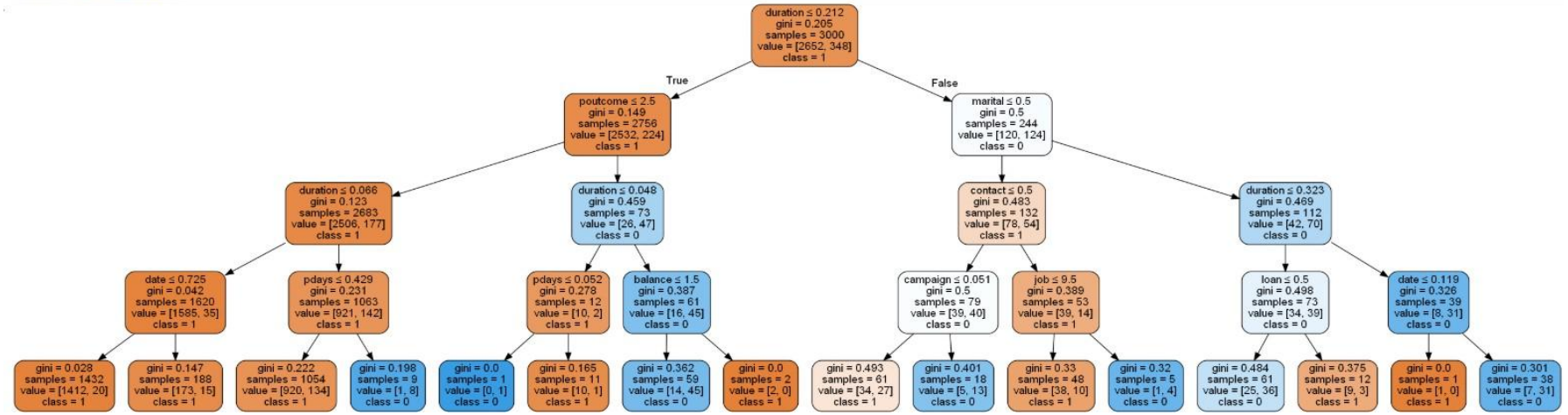
```
depth: 1 | 准确率: 0.8790786948176583
depth: 2 | 准确率: 0.8733205374280231
depth: 3 | 准确率: 0.8963531669865643
depth: 4 | 准确率: 0.9001919385796545
depth: 5 | 准确率: 0.9001919385796545
depth: 6 | 准确率: 0.8848368522072937
depth: 7 | 准确率: 0.8829174664107485
depth: 8 | 准确率: 0.8714011516314779
depth: 9 | 准确率: 0.8694817658349329
```

补充内容

⑤ 可视化决策树模型

```
from IPython.display import Image
from sklearn import tree
import pydotplus
import os
os.environ["PATH"] += os.pathsep + 'C:/Program Files/Graphviz 2.44.1/bin/'
dot_data = tree.export_graphviz(dtc, out_file=None,
                        feature_names=X_train.columns,
                        class_names=['1', '0'],
                        filled=True, rounded=True,
                        special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data)
graph.write_png("D:/大数据/实验二/DecisionTree.gif")

Image(graph.create_png())
```



大数据导论实验

同学们，请开始实验吧！