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

目录

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（10分）

Hadoop环境配置与基本操作

掌握大数据存储与检索的开源软件工具，并能完成给定的大数据存储与检索任务。

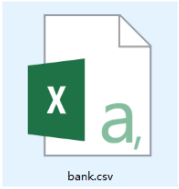
实验二（20分）

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；编码实现一个经典数据挖掘算法。

实验二任务

实验任务：提供银行精准营销解决方案



该数据集与葡萄牙银行机构的直接营销活动有关。这些营销活动是以电话为基础的。**营销活动的目的是希望客户能够认购自己的产品。**通常来说，银行机构的客服人员至少需要联系一次客户来得知客户是否将认购银行的产品（定期存款）。**案例数据是随机挑选的部分银行客户基本资料数据和与客户联系的相关数据**，包括年龄、婚姻状况、受教育程度、联系持续时间、联系次数等。

数据集	银行营销
记录数量	4521
属性	16
类别	2

因此，与该数据集对应的任务是分类任务，而分类目标是预测客户是 (yes) 否 (no) 认购定期存款（变量y）。

数据说明

客户基本资料的数据:

属性	描述	类型	缺失值
age	年龄	数值	无
job	工作类型	分类	unknown
marital	婚姻状况	分类	无
education	受教育程度	分类	unknown
default	是否存在拖欠	分类	无
balance	平均年度余额（欧元）	数值	无
housing	是否有住房贷款	分类	无
loan	是否有个人贷款	分类	无

数据说明

与客户联络的相关数据:

属性	描述	类型	缺失值
contact	联络方式	分类	unknown
day	最后一次联系是几号	数值	无
month	最后一次联系的月份	分类	无
duration	最后一次联系的持续时间（秒）	数值	无
campaign	此营销活动期间和此客户联系的次数，包括最后一次联系	数值	无
pdays	从上一个营销活动最后一次联系客户后经过的天数，-1表示之前未联系过客户	数值	无
previous	此营销活动之前和此客户联系的次数	数值	无
poutcome	上一次营销活动的结果	分类	unknown
y（目标）	客户是否将订购银行定期存款	分类	无

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
```

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 读取CSV文件，获取所有数据

```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt

df = pd.read_csv("D:/大数据/实验二/bank.csv")
cols = df.columns.values
cols = [col.replace('\"', '') for col in cols[0].split(';')]
print(cols)
```

```
['age', 'job', 'marital', 'education', 'default', 'balance', 'housing', 'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays', 'previous', 'poutcome', 'y']
```

```
datas = []
for i in range(len(df)):
    s = df.iloc[i].values[0]
    datas.append([item.replace('\"', '') for item in s.split(';')])
datas = np.array(datas)
print(datas)
```

```
[['30' 'unemployed' 'married' ... '0' 'unknown' 'no']
 ['33' 'services' 'married' ... '4' 'failure' 'no']
 ['35' 'management' 'single' ... '1' 'failure' 'no']
 ...
 ['57' 'technician' 'married' ... '0' 'unknown' 'no']
 ['28' 'blue-collar' 'married' ... '3' 'other' 'no']
 ['44' 'entrepreneur' 'single' ... '7' 'other' 'no']]
```

实验步骤

数据读取

② 转化为DataFrame格式

```
df_frame = {}
for i in range(len(cols)):
    df_frame[cols[i]] = datas[:,i]
df = pd.DataFrame(df_frame)
df
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0	unknown	no
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4	failure	no
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1	failure	no
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	unknown	no
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	0	unknown	no
5	35	management	single	tertiary	no	747	no	no	cellular	23	feb	141	2	176	3	failure	no
6	36	self-employed	married	tertiary	no	307	yes	no	cellular	14	may	341	1	330	2	other	no
7	39	technician	married	secondary	no	147	yes	no	cellular	6	may	151	2	-1	0	unknown	no
8	41	entrepreneur	married	tertiary	no	221	yes	no	unknown	14	may	57	2	-1	0	unknown	no
9	43	services	married	primary	no	-88	yes	yes	cellular	17	apr	313	1	147	2	failure	no
10	39	services	married	secondary	no	9374	yes	no	unknown	20	may	273	1	-1	0	unknown	no
11	43	admin.	married	secondary	no	264	yes	no	cellular	17	apr	113	2	-1	0	unknown	no
12	36	technician	married	tertiary	no	1109	no	no	cellular	13	aug	328	2	-1	0	unknown	no
13	20	student	single	secondary	no	502	no	no	cellular	30	apr	261	1	-1	0	unknown	yes
14	31	blue-collar	married	secondary	no	360	yes	yes	cellular	29	jan	89	1	241	1	failure	no
15	40	management	married	tertiary	no	194	no	yes	cellular	29	aug	189	2	-1	0	unknown	no

实验步骤

数据读取

③ 类型转换

将age、balance、duration、 campaign、pdays和previous六个字段从字符类型转为整数类型。

```
df[['age','balance','duration', 'campaign','pdays','previous']] = df[['age','balance','duration', 'campaign','pdays','previous']].astype(int)
df.dtypes
```

```
age           int32
job          object
marital      object
education    object
default      object
balance       int32
housing      object
loan         object
contact      object
day          object
month        object
duration      int32
campaign      int32
pdays         int32
previous      int32
poutcome     object
y            object
Length: 17, dtype: object
```

调用astype后返回的是一个新的对象，因此需要对原来的数据列进行赋值。之后再调用dtypes参数即可查看属性类型。

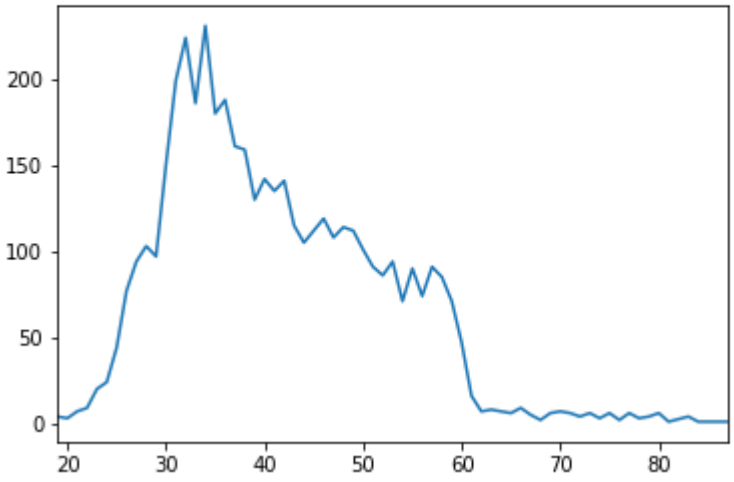
实验步骤

数据读取

④ 数据可视化

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

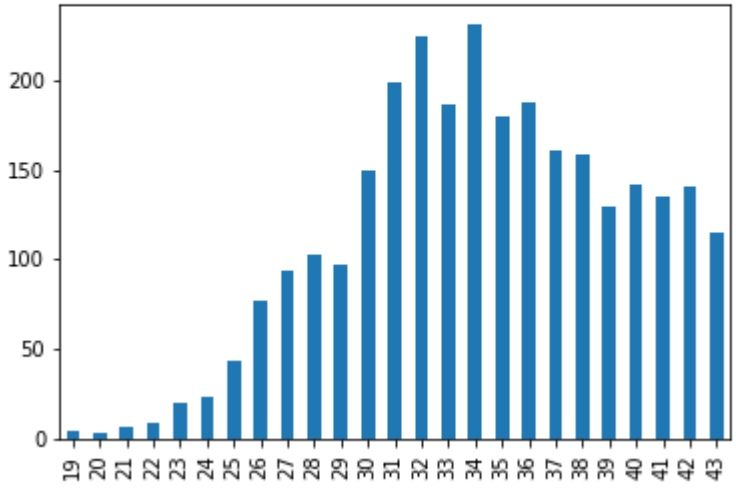
```
<matplotlib.axes._subplots.AxesSubplot at 0x247287b8be0>
```



折线图

```
df['age'].value_counts().sort_index().head(25).plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24729ad5898>
```



柱状图

实验步骤

数据预处理

① 筛选重复值并删除

```
df.duplicated()
```

```
0       False
1       False
2       False
3       False
4       False
        ...
4516    False
4517    False
4518    False
4519    False
4520    False
Length: 4521, dtype: bool
```

```
df.drop_duplicates()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0	unknown	no
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4	failure	no
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1	failure	no
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	unknown	no
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	0	unknown	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4516	33	services	married	secondary	no	-333	yes	no	cellular	30	jul	329	5	-1	0	unknown	no
4517	57	self-employed	married	tertiary	yes	-3313	yes	yes	unknown	9	may	153	1	-1	0	unknown	no
4518	57	technician	married	secondary	no	295	no	no	cellular	19	aug	151	11	-1	0	unknown	no
4519	28	blue-collar	married	secondary	no	1137	no	no	cellular	6	feb	129	4	211	3	other	no
4520	44	entrepreneur	single	tertiary	no	1136	yes	yes	cellular	3	apr	345	2	249	7	other	no

② 将yes/no分类映射为0/1分类

```
class_mapping = {'no':0, 'yes':1}
df['default'] = df['default'].map(class_mapping)
df['housing'] = df['housing'].map(class_mapping)
df['loan'] = df['loan'].map(class_mapping)
df['y'] = df['y'].map(class_mapping)
```

实验步骤

数据预处理

③ 将月份和日期转化成距离现在的天数，并删除原有的月份与日期

```
month_dict = {'oct': '10', 'may': '05', 'apr': '04', 'jun': '06', 'feb': '02', 'aug': '08',
              'jan': '01', 'jul': '07', 'nov': '11','sep': '09', 'mar': '03', 'dec': '12'}
df['month'] = df['month'].map(month_dict)
df['date'] = '2019' + '-' + df['month'] + '-' + df['day']
df['date'] = pd.to_datetime(df['date'], format="%Y-%m-%d")
df['date'] = pd.to_datetime('2020-01-01', format="%Y-%m-%d") - df['date']
df['date'] = df['date'].dt.days
print(df[['date']])
```

```
      date
0       74
1      235
2      260
3      212
4      241
...    ...
4516   155
4517   237
4518   135
4519   329
4520   273
```

```
del(df['day'])
del(df['month'])
```

实验步骤

2 数据预处理

④ 将字符型分类变量转化为数值型并处理缺失值

```
jobs = df['job'].unique()
job_mapping = {jobs[i]: i for i in range(jobs.shape[0])}

maritals =  df['marital'].unique()
marital_mapping = {maritals[i]: i for i in range(maritals.shape[0])}

educations =  df['education'].unique()
education_mapping = {educations[i]: i for i in range(educations.shape[0])}

contacts =  df['contact'].unique()
contact_mapping = {contacts[i]: i for i in range(contacts.shape[0])}

poutcomes = df['poutcome'].unique()
poutcome_mapping = {poutcomes[i]: i for i in range(poutcomes.shape[0])}

df['marital'] = df['marital'].map(marital_mapping)
df['job'] = df['job'].map(job_mapping)
df['education'] = df['education'].map(education_mapping)
df['contact'] = df['contact'].map(contact_mapping)
df['poutcome'] = df['poutcome'].map(poutcome_mapping)
df
```

	age	job	marital	education	default	balance	housing	loan	contact	duration	campaign	pdays	previous	poutcome	y	date
0	30	0	0	0	0	1787	0	0	0	79	1	-1	0	0	0	74
1	33	1	0	1	0	4789	1	1	0	220	1	339	4	1	0	235
2	35	2	1	2	0	1350	1	0	0	185	1	330	1	1	0	260
3	30	2	0	2	0	1476	1	1	1	199	4	-1	0	0	0	212
4	59	3	0	1	0	0	1	0	1	226	1	-1	0	0	0	241
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4516	33	1	0	1	0	-333	1	0	0	329	5	-1	0	0	0	155
4517	57	4	0	2	1	-3313	1	1	1	153	1	-1	0	0	0	237
4518	57	5	0	1	0	295	0	0	0	151	11	-1	0	0	0	135
4519	28	3	0	1	0	1137	0	0	0	129	4	211	3	2	0	329
4520	44	6	1	2	0	1136	1	1	0	345	2	249	7	2	0	273

4521 rows × 16 columns

实验步骤

数据预处理

⑤ 数据离散化

```
bins = [18, 25, 35, 45, 55, 100]   # 指定年龄的分界点
df['age'] = pd.cut(df['age'], bins, labels=False)
bins = [-np.inf, 4137.1, 11587.2, np.inf]
df['balance'] = pd.cut(df['balance'], bins, labels=False)
print(df[['age','balance']])
```

```
      age  balance
0       1        0
1       1        1
2       1        0
3       1        0
4       4        0
...   ...      ...
4516    1        0
4517    4        0
4518    4        0
4519    1        0
4520    2        0
```

分箱

实验步骤

数据预处理

⑥ 数据归一化

```
cols = ['pdays', 'duration', 'campaign','date']
for col in cols:
    df[col] = (df[col] - df[col].min()) / (df[col].max() - df[col].min())
print(df[['pdays', 'duration', 'campaign','date']])
```

```
         pdays  duration  campaign      date
0     0.000000  0.024826  0.000000  0.203911
1     0.389908  0.071500  0.000000  0.653631
2     0.379587  0.059914  0.000000  0.723464
3     0.000000  0.064548  0.061224  0.589385
4     0.000000  0.073486  0.000000  0.670391
...        ...       ...       ...       ...
4516  0.000000  0.107580  0.081633  0.430168
4517  0.000000  0.049321  0.000000  0.659218
4518  0.000000  0.048659  0.204082  0.374302
4519  0.243119  0.041377  0.061224  0.916201
4520  0.286697  0.112877  0.020408  0.759777

[4521 rows x 4 columns]
```

实验步骤

数据预处理

⑦ 保存数据

```
df.to_csv('D:/大数据/实验二/after_bank.csv')
df
```

	age	job	marital	education	default	balance	housing	loan	contact	duration	campaign	pdays	previous	poutcome	y	date
0	1	0	0	0	0	0	0	0	0	0.024826	0.000000	0.000000	0	0	0	0.203911
1	1	1	0	1	0	1	1	1	0	0.071500	0.000000	0.389908	4	1	0	0.653631
2	1	2	1	2	0	0	1	0	0	0.059914	0.000000	0.379587	1	1	0	0.723464
3	1	2	0	2	0	0	1	1	1	0.064548	0.061224	0.000000	0	0	0	0.589385
4	4	3	0	1	0	0	1	0	1	0.073486	0.000000	0.000000	0	0	0	0.670391
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4516	1	1	0	1	0	0	1	0	0	0.107580	0.081633	0.000000	0	0	0	0.430168
4517	4	4	0	2	1	0	1	1	1	0.049321	0.000000	0.000000	0	0	0	0.659218
4518	4	5	0	1	0	0	0	0	0	0.048659	0.204082	0.000000	0	0	0	0.374302
4519	1	3	0	1	0	0	0	0	0	0.041377	0.061224	0.243119	3	2	0	0.916201
4520	2	6	1	2	0	0	1	1	0	0.112877	0.020408	0.286697	7	2	0	0.759777

4521 rows × 16 columns

实验步骤

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]
```

实验步骤

构建决策树

③ 训练模型

决策树是一种树形结构的**分类器**，通过顺序询问分类点的属性决定分类点最终类别。通常根据特征的信息增益或其他指标，构建一棵决策树。

```
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
```

实验步骤

构建决策树

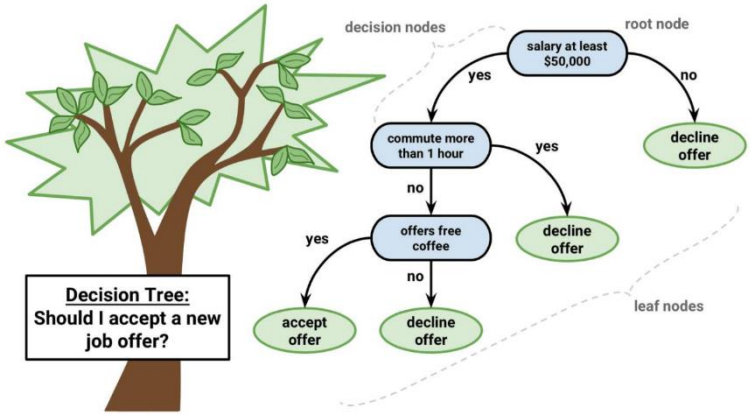
④ 参数调整

构建模型中很重要的一步是**调参**。在sklearn中，模型的参数是通过方法参数来决定的，以下给出sklearn中，决策树的参数：

```
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)
```

实验步骤

构建决策树



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")。指定分裂节点时的策略。
(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：最大深度，防止过拟合
int or None, optional (default=None)。指定树的最大深度。
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)。表示分裂一个内部节点需要的做少样本数。
(1).如果为整数，则min_samples_split就是最少样本数。
(2).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)

实验步骤

构建决策树

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
```

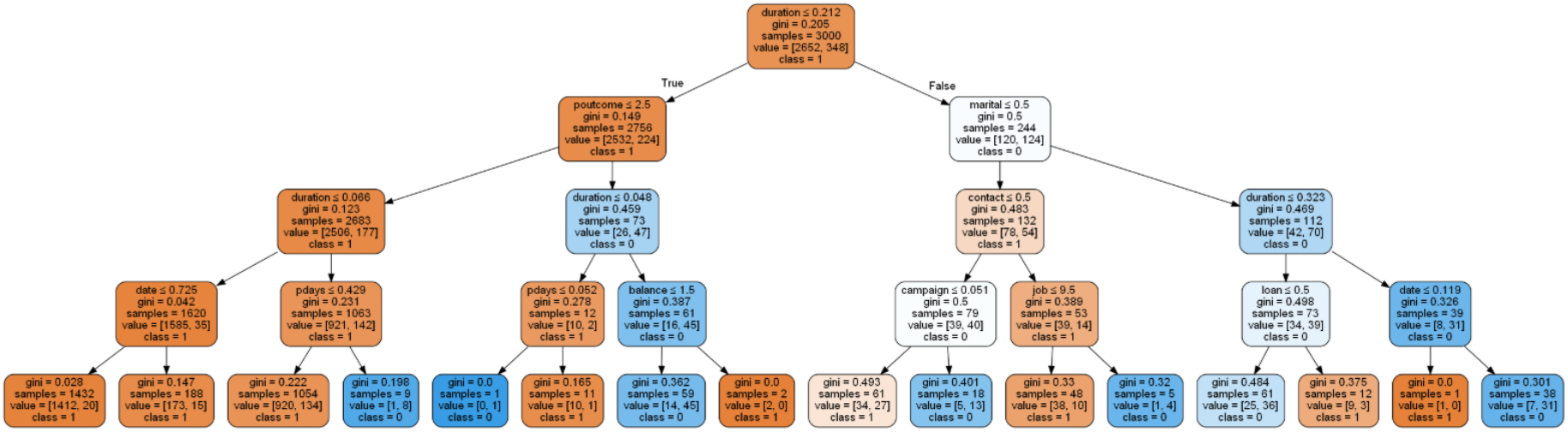
实验步骤

构建决策树

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())
```



思考题

1. 请使用sklearn中的模型调参利器网格搜索(GridSearchCV) 寻找本实验中决策树模型的最优参数和结果。

2. 请构建sklearn中的其他分类模型，并比较不同分类器的结果。

注：思考题有10%的加分， 但总分不超过该次实验满分。

大数据导论实验

同学们，请开始实验吧！