用户分层

一、课前准备

- 熟悉python的基本用法
- 熟悉决策树的使用

二、课堂主题

- RFM用户分层
- 决策树用户分层

三、课堂目标

- 了解什么是RFM
- RFM用户分层项目制作
- 决策树用户分层项目制作

四、知识要点

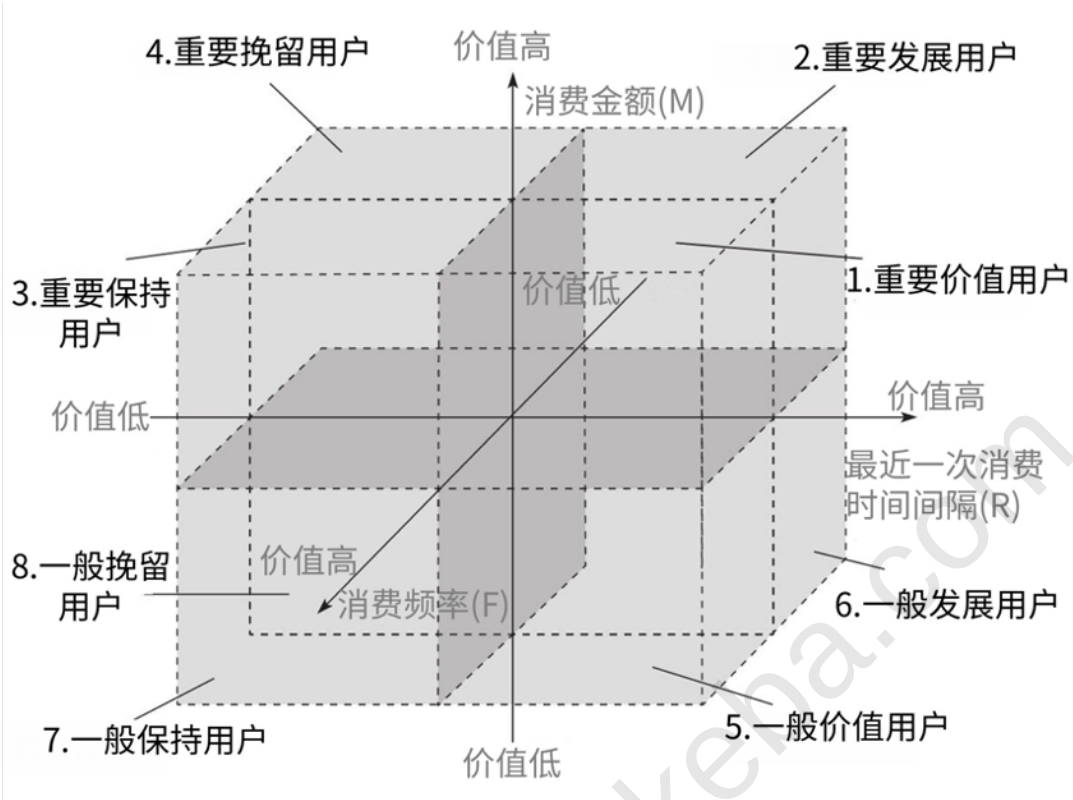
4.1 什么是RFM

RFM模型是衡量客户价值和客户创利能力的重要工具和手段。在众多的客户关系管理(CRM)的分析模式中，RFM模型是被广泛提到的。该机械模型通过一个客户的近期购买行为、购买的总体频率以及花了多少钱3项指标来描述该客户的价值状况。

R-Recency（近期）：最近一次消费是多久以前；R越大，表示客户交易发生的日期越久，反之则表示客户交易发生的日期越近。

F-Frequency（频率）：购买频率：F值越大，表示客户交易越频繁，反之则表示客户交易不够活跃。

M-Monetary（消费）：设定时间段内客户的总消费金额；M值越大，表示客户价值越高，反正则表示客户价值越低，这是衡量客户价值的最重要的指标。



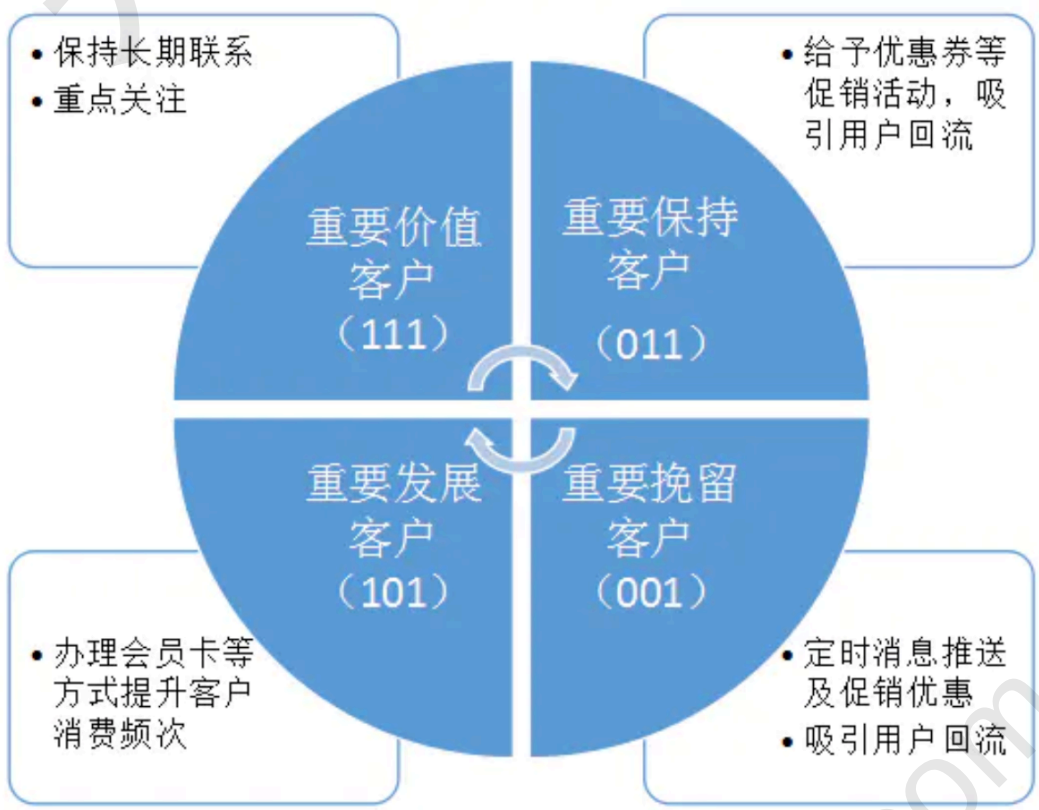
4.2 常见的用户问题

- 谁是我的重要价值客户，他们有什么特点?
- 谁是我需要重点保持联系的客户，他们有什么特点?
- 谁是我的重要发展客户，他们都有什么特点?
- 谁是我的重要挽留客户，他们都有什么特点?

基于RFM的客户分类表（以RFM为XYZ轴，1代表高，0代表低）

R（时间间隔）	F（消费频率）	M（消费金额）	客户类型
1	1	1	重要价值客户
0	1	1	重要保持客户
1	0	1	重要发展客户
0	0	1	重要挽留客户
1	1	0	一般价值客户
1	0	0	一般发展客户
0	1	0	一般保持客户
0	0	0	一般挽留客户

- 重要价值客户（111）：最近消费时间较短、消费频次和消费金额都较高。这是门店应该主要关注的VIP客户。
- 重要保持客户（011）：最近消费时间较长，消费频次和消费金额都较高。说明这是个一段时间没来的忠实客户，我们需要主动和他保持联系。
- 重要发展客户（101）：最近消费时间较短、消费金额高，但消费频次较低。忠诚度不高，很有潜力的用户，必须重点发展。
- 重要挽留客户（001）：最近消费时间较长、消费频次不高，但消费金额高的用户，可能是将要流失或者已经要流失的用户，应当吸引客户回流。



4.3 RFM电商用户分层

- 字段说明:

属性名称	属性说明
Row ID	行编号
Order ID	订单ID
Order Date	订单日期
Ship Date	发货日期
Ship Mode	发货模式
Customer ID	客户ID
Customer Name	客户姓名
Segment	客户类别
City	客户所在城市
State	客户城市所在州
Country	客户所在国家
Postal Code	邮编
Market	商店所属区域
Region	商店所属洲
Product ID	产品ID
Category	产品类别
Sub-Category	产品子类别
Product Name	产品名称
Sales	销售额
Quantity	销售量
Discount	折扣
Profit	利润
Shipping Cost	发货成本
Order Priority	订单优先级

- 数据清洗
  1.缺失值
  2.异常值
  3.重复值
- 读取数据

```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")

data = pd.read_csv('./dataset.csv', encoding='ISO-8859-1')
data.head()

# x y:no
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	RowID	OrderID	OrderDate	ShipDate	ShipMode	CustomerID	CustomerName	Segment	City	State
0	1	IN-2011-47883	2011/1/1	2011/1/8	Standard Class	JH-15985	Joseph Holt	Consumer	Wagga Wagga	New South Wales
1	2	IN-2011-47883	2011/1/1	2011/1/8	Standard Class	JH-15985	Joseph Holt	Consumer	Wagga Wagga	New South Wales
2	3	IN-2011-47883	2011/1/1	2011/1/8	Standard Class	JH-15985	Joseph Holt	Consumer	Wagga Wagga	New South Wales
3	4	IT-2011-3647632	2011/1/1	2011/1/5	Second Class	EM-14140	Eugene Moren	Home Office	Stockholm	Stockholm
4	5	HU-2011-1220	2011/1/1	2011/1/5	Second Class	AT-735	Annie Thurman	Consumer	Budapest	Budapest

5 rows × 24 columns

- 数据清洗

```
# 根据业务需要提取数据
# shipdata发货日期
# orderdata下单日期
data['ShipDate'] = pd.to_datetime(data['ShipDate'])
data['OrderDate'] = pd.to_datetime(data['OrderDate'])
# 时间间隔
data['interval'] = (data.ShipDate - data.OrderDate).dt.total_seconds()
data[data.interval < 0]
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	RowID	OrderID	OrderDate	ShipDate	ShipMode	CustomerID	CustomerName	Segment	City	State	...	Category
14	15	ES-2011-4869686	2011-01-03	2011-01-01	Standard Class	DD-13570	Dorothy Dickinson	Consumer	Southport	England	...	Furniture
490	490	TU-2011-7850	2011-02-05	2011-01-12	Standard Class	AW-840	Anthony Witt	Consumer	Gaziantep	Gaziantep	...	Furniture
20868	20867	ES-2013-5588419	2013-02-14	2013-01-19	Second Class	RF-19840	Roy Französisch	Consumer	Solihull	England	...	Office Supplies
35672	35670	TU-2014-9870	2014-03-13	2014-01-20	Standard Class	QJ-9255	Quincy Jones	Corporate	Izmir	Izmir	...	Office Supplies

4 rows × 25 columns

```
data.drop(index=data[data.interval < 0].index, inplace=True)
data
data['interval'] = data.ShipDate - data.OrderDate
data.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	RowID	OrderID	OrderDate	ShipDate	ShipMode	CustomerID	CustomerName	Segment	City	State	...	Category	C
0	1	IN-2011-47883	2011-01-01	2011-01-08	Standard Class	JH-15985	Joseph Holt	Consumer	Wagga Wagga	New South Wales	...	Office Supplies	Sup
1	2	IN-2011-47883	2011-01-01	2011-01-08	Standard Class	JH-15985	Joseph Holt	Consumer	Wagga Wagga	New South Wales	...	Office Supplies	Pap
2	3	IN-2011-47883	2011-01-01	2011-01-08	Standard Class	JH-15985	Joseph Holt	Consumer	Wagga Wagga	New South Wales	...	Furniture	Fur
3	4	IT-2011-3647632	2011-01-01	2011-01-05	Second Class	EM-14140	Eugene Moren	Home Office	Stockholm	Stockholm	...	Office Supplies	Pap
4	5	HU-2011-1220	2011-01-01	2011-01-05	Second Class	AT-735	Annie Thurman	Consumer	Budapest	Budapest	...	Office Supplies	Sto

5 rows × 25 columns

```
# 提取数据时,处理与业务流程不符合数据,售价为负
data[data.Sales < 0]
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	RowID	OrderID	OrderDate	ShipDate	ShipMode	CustomerID	CustomerName	Segment	City	State	...	Category	Sub-Category	Pr

0 rows × 25 columns

```
data.shape
```

```
(51097, 25)
```

```
data.count()
```

```
RowID            51097
OrderID          51097
OrderDate        51097
ShipDate         51097
ShipMode         51086
CustomerID       51097
CustomerName     51097
Segment          51097
City             51097
State            51097
Country          51097
PostalCode        9962
Market           51097
Region           51097
ProductID        51097
Category         51097
Sub-Category     51097
ProductName      51097
Sales            51097
Quantity         51097
Discount         51097
Profit           51097
ShippingCost     51097
OrderPriority    51097
interval         51097
dtype: int64
```

```
data.isna().sum()
```

```
RowID               0
OrderID             0
OrderDate           0
ShipDate            0
ShipMode           11
CustomerID          0
CustomerName        0
Segment             0
City                0
State               0
Country             0
PostalCode      41135
Market              0
Region              0
ProductID           0
Category            0
Sub-Category        0
ProductName         0
Sales               0
Quantity            0
Discount            0
Profit              0
ShippingCost        0
OrderPriority       0
interval            0
dtype: int64
```

```
data.describe()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	RowID	PostalCode	Sales	Quantity	Discount	Profit	ShippingCost	interval
count	51097.000000	9962.000000	51097.000000	51097.000000	51097.000000	51097.000000	51097.000000	51097
mean	25549.751668	55225.189319	246.568189	3.476036	0.143456	28.515431	26.385308	3 days 23:15:23.302737
std	14750.670508	32062.973837	487.774029	2.279122	0.213386	174.483766	57.286297	1 days 17:30:17.616193
min	1.000000	1040.000000	0.444000	1.000000	0.000000	-6599.978000	0.000000	0 days 00:00:00
25%	12775.000000	23223.000000	30.816000	2.000000	0.000000	0.000000	2.610000	3 days 00:00:00
50%	25550.000000	57103.000000	85.140000	3.000000	0.000000	9.240000	7.800000	4 days 00:00:00
75%	38324.000000	90008.000000	251.100000	5.000000	0.200000	36.810000	24.450000	5 days 00:00:00
max	51098.000000	99301.000000	22638.480000	14.000000	1.500000	8399.976000	933.570000	7 days 00:00:00

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51097 entries, 0 to 51100
Data columns (total 25 columns):
RowID            51097 non-null int64
OrderID          51097 non-null object
OrderDate        51097 non-null datetime64[ns]
ShipDate         51097 non-null datetime64[ns]
ShipMode         51086 non-null object
CustomerID       51097 non-null object
CustomerName     51097 non-null object
Segment          51097 non-null object
City             51097 non-null object
State            51097 non-null object
Country          51097 non-null object
PostalCode       9962 non-null float64
Market           51097 non-null object
Region           51097 non-null object
ProductID        51097 non-null object
Category         51097 non-null object
Sub-Category     51097 non-null object
ProductName      51097 non-null object
Sales            51097 non-null float64
Quantity         51097 non-null int64
Discount         51097 non-null float64
Profit           51097 non-null float64
ShippingCost     51097 non-null float64
OrderPriority    51097 non-null object
interval         51097 non-null timedelta64[ns]
dtypes: datetime64[ns](2), float64(5), int64(2), object(15), timedelta64[ns](1)
memory usage: 10.1+ MB
```

```
# 查看行列数量
data.shape
# 查看各个列的非空数据量
data.count()  # count 计算非NAN 值的数量， 按照字段计算
#NAN统计
data.isna().sum()
#数据整体描述
data.describe()
#数据信息
data.info()
# 简单浏览下数据
data.head() # qian5
data.tail() # hou5
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51097 entries, 0 to 51100
Data columns (total 25 columns):
RowID            51097 non-null int64
OrderID          51097 non-null object
OrderDate        51097 non-null datetime64[ns]
ShipDate         51097 non-null datetime64[ns]
ShipMode         51086 non-null object
CustomerID       51097 non-null object
```

```
CustomerName     51097 non-null object
Segment          51097 non-null object
City             51097 non-null object
State            51097 non-null object
Country          51097 non-null object
PostalCode       9962 non-null float64
Market           51097 non-null object
Region           51097 non-null object
ProductID        51097 non-null object
Category         51097 non-null object
Sub-Category     51097 non-null object
ProductName      51097 non-null object
Sales            51097 non-null float64
Quantity         51097 non-null int64
Discount         51097 non-null float64
Profit           51097 non-null float64
ShippingCost     51097 non-null float64
OrderPriority    51097 non-null object
interval         51097 non-null timedelta64[ns]
dtypes: datetime64[ns](2), float64(5), int64(2), object(15), timedelta64[ns](1)
memory usage: 10.1+ MB
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	RowID	OrderID	OrderDate	ShipDate	ShipMode	CustomerID	CustomerName	Segment	City	State	...	Category
51096	51094	IN-2014-75603	2014-12-31	2015-01-05	Second Class	BS-11365	Bill Shonely	Corporate	Vijayawada	Andhra Pradesh	...	Office Supplies
51097	51095	TU-2014-5170	2014-12-31	2015-01-04	Second Class	VD-11670	Valerie Dominguez	Consumer	Konya	Konya	...	Furniture
51098	51096	MO-2014-2560	2014-12-31	2015-01-05	Standard Class	LP-7095	Liz Preis	Consumer	Agadir	Souss-M	...	Office Supplies
51099	51097	ES-2014-4785777	2014-12-31	2015-01-04	Standard Class	DP-13390	Dennis Pardue	Home Office	Hamburg	Hamburg	...	Office Supplies
51100	51098	CA-2014-143259	2014-12-31	2015-01-04	Standard Class	PO-18865	Patrick O'Donnell	Consumer	New York City	New York	...	Office Supplies

5 rows × 25 columns

object(label encoder) int float bool
sum()
groupby()

```
data.RowID.unique().size
# 重复查看
data[data.RowID.duplicated()]
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	RowID	OrderID	OrderDate	ShipDate	ShipMode	CustomerID	CustomerName	Segment	City	State	...	Category
141	141	ES-2011-4359424	2011-01-11	2011-01-15	Standard Class	DD-13570	Dorothy Dickinson	Consumer	Murcia	Murcia	...	Office Supplies
11566	11565	IN-2012-81581	2012-05-12	2012-05-16	Standard Class	BW-11200	Ben Wallace	Consumer	Canberra	Australian Capital Territory	...	Office Supplies
32648	32646	CM-2013-4490	2013-12-12	2013-12-14	Second Class	CW-1905	Carl Weiss	Home Office	Bamenda	Nord-Ouest	...	Office Supplies

3 rows × 25 columns

```
# 清洗RowID（行编号）
# RowID不重复的个数
# drop_duplicated()
data.RowID.unique().size
data[data.RowID.duplicated()]
data.drop(index=data[data.RowID.duplicated()].index, inplace=True)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51094 entries, 0 to 51100
Data columns (total 25 columns):
RowID             51094 non-null int64
OrderID           51094 non-null object
OrderDate         51094 non-null datetime64[ns]
ShipDate          51094 non-null datetime64[ns]
ShipMode          51083 non-null object
CustomerID        51094 non-null object
CustomerName      51094 non-null object
Segment           51094 non-null object
City              51094 non-null object
State             51094 non-null object
Country           51094 non-null object
PostalCode        9962 non-null float64
Market            51094 non-null object
Region            51094 non-null object
ProductID         51094 non-null object
Category          51094 non-null object
Sub-Category      51094 non-null object
ProductName       51094 non-null object
Sales             51094 non-null float64
Quantity          51094 non-null int64
Discount          51094 non-null float64
Profit            51094 non-null float64
ShippingCost      51094 non-null float64
OrderPriority     51094 non-null object
interval          51094 non-null timedelta64[ns]
dtypes: datetime64[ns](2), float64(5), int64(2), object(15), timedelta64[ns](1)
memory usage: 10.1+ MB
```

```
data[data.ShipMode.isnull()]
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	RowID	OrderID	OrderDate	ShipDate	ShipMode	CustomerID	CustomerName	Segment	City	State	...	Category
13666	13665	BU-2012-460	2012-07-21	2012-07-25	NaN	AH-30	Aaron Hawkins	Corporate	Pazardzhik	Pazardzhik	...	Office Supplies
13674	13673	BU-2012-460	2012-07-21	2012-07-25	NaN	AH-30	Aaron Hawkins	Corporate	Pazardzhik	Pazardzhik	...	Furniture
18385	18384	AG-2012-6390	2012-11-26	2012-11-30	NaN	GH-4410	Gary Hansen	Home Office	Algiers	Alger	...	Office Supplies
33167	33165	MZ-2013-7330	2013-12-23	2013-12-25	NaN	CS-1950	Carlos Soltero	Consumer	Chimoio	Manica	...	Office Supplies
33619	33617	AE-2013-1530	2013-12-31	2014-01-03	NaN	MY-7380	Maribeth Yedwab	Corporate	Ras al Khaymah	Ra's Al Khaymah	...	Office Supplies
33622	33620	AE-2013-1530	2013-12-31	2014-01-03	NaN	MY-7380	Maribeth Yedwab	Corporate	Ras al Khaymah	Ra's Al Khaymah	...	Office Supplies
38164	38162	MX-2014-153269	2014-05-19	2014-05-22	NaN	MC-17275	Marc Crier	Consumer	Rosario	Santa Fe	...	Office Supplies
44243	44241	AJ-2014-6170	2014-09-16	2014-09-18	NaN	SH-9975	Sally Hughsby	Corporate	Baku	Baki	...	Office Supplies
44321	44319	ES-2014-4138124	2014-09-17	2014-09-22	NaN	BF-11275	Beth Fritzler	Corporate	Oslo	Oslo	...	Technology
49931	49929	US-2014-134292	2014-12-14	2014-12-20	NaN	JK-15640	Jim Kriz	Home Office	La Chorrera	Panama	...	Office Supplies
50866	50864	IN-2014-64501	2014-12-29	2015-01-02	NaN	TB-21250	Tim Brockman	Consumer	Kanpur	Uttar Pradesh	...	Office Supplies

11 rows × 25 columns

```
# 清洗ShipMode
# 1.查看ShipMode空值
data[data.ShipMode.isnull()]
# 2.对空值进行修补
data['ShipMode'].fillna(value=data.ShipMode.mode()[0], inplace=True)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51094 entries, 0 to 51100
Data columns (total 25 columns):
RowID            51094 non-null int64
OrderID          51094 non-null object
OrderDate        51094 non-null datetime64[ns]
ShipDate         51094 non-null datetime64[ns]
ShipMode         51094 non-null object
CustomerID       51094 non-null object
CustomerName     51094 non-null object
Segment          51094 non-null object
```

```
City              51094 non-null object
State             51094 non-null object
Country           51094 non-null object
PostalCode        9962 non-null float64
Market            51094 non-null object
Region            51094 non-null object
ProductID         51094 non-null object
Category          51094 non-null object
Sub-Category      51094 non-null object
ProductName       51094 non-null object
Sales             51094 non-null float64
Quantity          51094 non-null int64
Discount          51094 non-null float64
Profit            51094 non-null float64
ShippingCost      51094 non-null float64
OrderPriority     51094 non-null object
interval          51094 non-null timedelta64[ns]
dtypes: datetime64[ns](2), float64(5), int64(2), object(15), timedelta64[ns](1)
memory usage: 10.1+ MB
```

```
data[data.Discount > 1]
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	RowID	OrderID	OrderDate	ShipDate	ShipMode	CustomerID	CustomerName	Segment	City	State	...	Category
7441	7441	TU-2011-3230	2011-11-23	2011-11-27	Standard Class	CS-1845	Cari Sayre	Corporate	Istanbul	Istanbul	...	Furniture
13913	13912	ID-2012-34884	2012-08-02	2012-08-08	Standard Class	SC-20305	Sean Christensen	Consumer	Adelaide	South Australia	...	Office Supplies
19195	19194	MX-2012-161704	2012-12-15	2012-12-20	Second Class	AH-10120	Adrian Hane	Home Office	Guadalajara	Jalisco	...	Furniture
22326	22325	ID-2013-34709	2013-04-15	2013-04-21	Standard Class	LC-16870	Lena Cacioppo	Consumer	Melbourne	Victoria	...	Office Supplies
22407	22406	IN-2013-25245	2013-04-17	2013-04-22	Standard Class	JH-15985	Joseph Holt	Consumer	Manila	National Capital	...	Furniture
22545	22544	IT-2013-3005581	2013-04-23	2013-04-30	Standard Class	AJ-10780	Anthony Jacobs	Corporate	The Hague	South Holland	...	Technology
23328	23327	ES-2013-5311844	2013-05-18	2013-05-19	First Class	JS-15880	John Stevenson	Consumer	Oslo	Oslo	...	Office Supplies
23733	23732	CA-2013-152730	2013-05-31	2013-06-05	Standard Class	EM-14140	Eugene Moren	Home Office	Superior	Wisconsin	...	Office Supplies
27082	27081	MO-2013-5960	2013-08-20	2013-08-22	Second Class	JG-5115	Jack Garza	Consumer	Tangier	Tanger-Tétouan	...	Office Supplies
34025	34023	CA-2014-127306	2014-01-15	2014-01-19	Standard Class	BH-11710	Brosina Hoffman	Consumer	Johnson City	Tennessee	...	Office Supplies
38856	38854	CA-2014-140872	2014-06-04	2014-06-11	Standard Class	NR-18550	Nick Radford	Consumer	Pembroke Pines	Florida	...	Office Supplies
41634	41632	ID-2014-62289	2014-08-03	2014-08-07	Standard Class	RB-19645	Robert Barroso	Corporate	Manila	National Capital	...	Technology
47274	47272	CA-2014-106691	2014-11-07	2014-11-13	Standard Class	CC-12370	Christopher Conant	Consumer	Houston	Texas	...	Office Supplies

13 rows × 25 columns

```
data[data.Discount < 0]
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	RowID	OrderID	OrderDate	ShipDate	ShipMode	CustomerID	CustomerName	Segment	City	State	...	Category	Sub-Category	Pr

0 rows × 25 columns

```
# 清洗Discount（折扣）

data[data.Discount > 1]
data[data.Discount < 0]
data['Discount'] = data['Discount'].mask(data['Discount'] > 1, None)
data
# 1.查看Discount空值
data[data.Discount.isnull()]
# 2.平均折扣
meanDiscount = round(
    data[data.Discount.notnull()].Discount.sum() /
    data[data.Discount.notnull()].Discount.size, 2)
meanDiscount
# 3.对Discount的数据进行填补
data['Discount'].fillna(value=meanDiscount, inplace=True)
data.info()
data
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51094 entries, 0 to 51100
Data columns (total 25 columns):
RowID            51094 non-null int64
OrderID          51094 non-null object
OrderDate        51094 non-null datetime64[ns]
ShipDate         51094 non-null datetime64[ns]
ShipMode         51094 non-null object
CustomerID       51094 non-null object
CustomerName     51094 non-null object
Segment          51094 non-null object
City             51094 non-null object
State            51094 non-null object
Country          51094 non-null object
PostalCode       9962 non-null float64
Market           51094 non-null object
Region           51094 non-null object
ProductID        51094 non-null object
Category         51094 non-null object
Sub-Category     51094 non-null object
ProductName      51094 non-null object
Sales            51094 non-null float64
Quantity         51094 non-null int64
Discount         51094 non-null float64
Profit           51094 non-null float64
ShippingCost     51094 non-null float64
OrderPriority    51094 non-null object
interval         51094 non-null timedelta64[ns]
dtypes: datetime64[ns](2), float64(5), int64(2), object(15), timedelta64[ns](1)
memory usage: 10.1+ MB
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	RowID	OrderID	OrderDate	ShipDate	ShipMode	CustomerID	CustomerName	Segment	City	State	...	Category
0	1	IN-2011-47883	2011-01-01	2011-01-08	Standard Class	JH-15985	Joseph Holt	Consumer	Wagga Wagga	New South Wales	...	Office Supplies
1	2	IN-2011-47883	2011-01-01	2011-01-08	Standard Class	JH-15985	Joseph Holt	Consumer	Wagga Wagga	New South Wales	...	Office Supplies
2	3	IN-2011-47883	2011-01-01	2011-01-08	Standard Class	JH-15985	Joseph Holt	Consumer	Wagga Wagga	New South Wales	...	Furniture
3	4	IT-2011-3647632	2011-01-01	2011-01-05	Second Class	EM-14140	Eugene Moren	Home Office	Stockholm	Stockholm	...	Office Supplies
4	5	HU-2011-1220	2011-01-01	2011-01-05	Second Class	AT-735	Annie Thurman	Consumer	Budapest	Budapest	...	Office Supplies
...	...	...	...	...	...	...	...	...	...	...	...	...
51096	51094	IN-2014-75603	2014-12-31	2015-01-05	Second Class	BS-11365	Bill Shonely	Corporate	Vijayawada	Andhra Pradesh	...	Office Supplies
51097	51095	TU-2014-5170	2014-12-31	2015-01-04	Second Class	VD-11670	Valerie Dominguez	Consumer	Konya	Konya	...	Furniture
51098	51096	MO-2014-2560	2014-12-31	2015-01-05	Standard Class	LP-7095	Liz Preis	Consumer	Agadir	Souss-M	...	Office Supplies
51099	51097	ES-2014-4785777	2014-12-31	2015-01-04	Standard Class	DP-13390	Dennis Pardue	Home Office	Hamburg	Hamburg	...	Office Supplies
51100	51098	CA-2014-143259	2014-12-31	2015-01-04	Standard Class	PO-18865	Patrick O'Donnell	Consumer	New York City	New York	...	Office Supplies

51094 rows × 25 columns

```
# 清洗PostalCode列
# 实际情况去考虑
data.drop(columns=['PostalCode'], inplace=True)
data
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	RowID	OrderID	OrderDate	ShipDate	ShipMode	CustomerID	CustomerName	Segment	City	State	...	Category
0	1	IN-2011-47883	2011-01-01	2011-01-08	Standard Class	JH-15985	Joseph Holt	Consumer	Wagga Wagga	New South Wales	...	Office Supplies
1	2	IN-2011-47883	2011-01-01	2011-01-08	Standard Class	JH-15985	Joseph Holt	Consumer	Wagga Wagga	New South Wales	...	Office Supplies
2	3	IN-2011-47883	2011-01-01	2011-01-08	Standard Class	JH-15985	Joseph Holt	Consumer	Wagga Wagga	New South Wales	...	Furniture
3	4	IT-2011-3647632	2011-01-01	2011-01-05	Second Class	EM-14140	Eugene Moren	Home Office	Stockholm	Stockholm	...	Office Supplies
4	5	HU-2011-1220	2011-01-01	2011-01-05	Second Class	AT-735	Annie Thurman	Consumer	Budapest	Budapest	...	Office Supplies
...	...	...	...	...	...	...	...	...	...	...	...	...
51096	51094	IN-2014-75603	2014-12-31	2015-01-05	Second Class	BS-11365	Bill Shonely	Corporate	Vijayawada	Andhra Pradesh	...	Office Supplies
51097	51095	TU-2014-5170	2014-12-31	2015-01-04	Second Class	VD-11670	Valerie Dominguez	Consumer	Konya	Konya	...	Furniture
51098	51096	MO-2014-2560	2014-12-31	2015-01-05	Standard Class	LP-7095	Liz Preis	Consumer	Agadir	Souss-M	...	Office Supplies
51099	51097	ES-2014-4785777	2014-12-31	2015-01-04	Standard Class	DP-13390	Dennis Pardue	Home Office	Hamburg	Hamburg	...	Office Supplies
51100	51098	CA-2014-143259	2014-12-31	2015-01-04	Standard Class	PO-18865	Patrick O'Donnell	Consumer	New York City	New York	...	Office Supplies

51094 rows × 24 columns

```
# 填充数据L：用算法：预测 线性回归（共线性的特征，去填充）
```

```
# 数据整理
# 根据OrderData生成 年份\月份\季度的字段
data['Order-year'] = data['OrderDate'].dt.year
data['Order-month'] = data['OrderDate'].dt.month
data['quarter'] = data['OrderDate'].dt.to_period('Q')
result = data[['OrderDate', 'Order-year', 'Order-month', 'quarter']].head()
print(result)
```

```
   OrderDate  Order-year  Order-month quarter
0 2011-01-01        2011            1  2011Q1
1 2011-01-01        2011            1  2011Q1
2 2011-01-01        2011            1  2011Q1
3 2011-01-01        2011            1  2011Q1
4 2011-01-01        2011            1  2011Q1
```

```
# 每年销售额的增长情况
sales_year = data.groupby(by='Order-year')['Sales'].sum()
print(sales_year)

sales_rate_12 = sales_year[2012] / sales_year[2011] - 1
sales_rate_13 = sales_year[2013] / sales_year[2012] - 1
sales_rate_14 = sales_year[2014] / sales_year[2013] - 1
print(sales_rate_12, sales_rate_13, sales_rate_14)
```

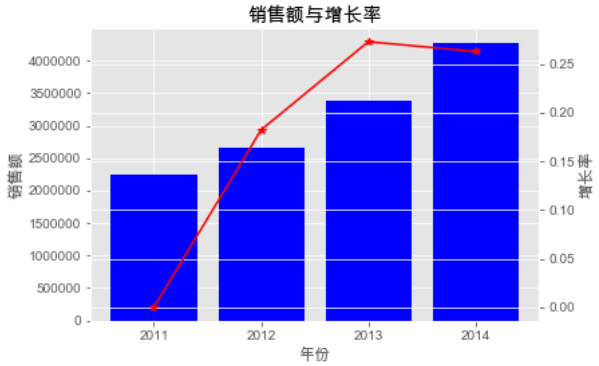
```
sales_rate_12_label = "%.2f%%" % (sales_rate_12 * 100)
sales_rate_13_label = "%.2f%%" % (sales_rate_13 * 100)
sales_rate_14_label = "%.2f%%" % (sales_rate_14 * 100)
print(sales_rate_12_label, sales_rate_13_label, sales_rate_14_label)

sales_rate = pd.DataFrame({
    'sales_all':
    sales_year,
    'sales_rate': [0, sales_rate_12, sales_rate_13, sales_rate_14],
    'sales_rate_label':
    ['0.00%', sales_rate_12_label, sales_rate_13_label, sales_rate_14_label]
})
print(sales_rate)
```

```
Order-year
2011    2.254364e+06
2012    2.665397e+06
2013    3.393116e+06
2014    4.285524e+06
Name: Sales, dtype: float64
0.18232782910632062 0.2730243556885772 0.2630055018020905
18.23% 27.30% 26.30%
               sales_all  sales_rate sales_rate_label
Order-year
2011        2.254364e+06    0.000000            0.00%
2012        2.665397e+06    0.182328           18.23%
2013        3.393116e+06    0.273024           27.30%
2014        4.285524e+06    0.263006           26.30%
```

```
# 每年销售额的增长情况
sales_year = data.groupby(by='Order-year')['Sales'].sum()
print(sales_year)

sales_rate_12 = sales_year[2012] / sales_year[2011] - 1
sales_rate_13 = sales_year[2013] / sales_year[2012] - 1
sales_rate_14 = sales_year[2014] / sales_year[2013] - 1
print(sales_rate_12, sales_rate_13, sales_rate_14)

sales_rate_12_label = "%.2f%%" % (sales_rate_12 * 100)
sales_rate_13_label = "%.2f%%" % (sales_rate_13 * 100)
sales_rate_14_label = "%.2f%%" % (sales_rate_14 * 100)
print(sales_rate_12_label, sales_rate_13_label, sales_rate_14_label)

sales_rate = pd.DataFrame({
    'sales_all':
    sales_year,
    'sales_rate': [0, sales_rate_12, sales_rate_13, sales_rate_14],
    'sales_rate_label':
    ['0.00%', sales_rate_12_label, sales_rate_13_label, sales_rate_14_label]
})
print(sales_rate)

import matplotlib.pyplot as plt
import matplotlib as mpl
# 设置字体
mpl.rcParams['font.sans-serif'] = ['SimHei']     # win
plt.rcParams["font.family"] = 'Arial Unicode MS' # mac
# 设置风格
plt.style.use('ggplot')
sales_rate = pd.DataFrame({
    'sales_all':
    sales_year,
    'sales_rate': [0, sales_rate_12, sales_rate_13, sales_rate_14],
    'sales_rate_label':
    ['0.00%', sales_rate_12_label, sales_rate_13_label, sales_rate_14_label]
})
y1 = sales_rate['sales_all']
y2 = sales_rate['sales_rate']
x = [str(value) for value in sales_rate.index.tolist()]
# 新建figure对象
fig = plt.figure()
# 新建子图1
ax1 = fig.add_subplot(1, 1, 1)
# ax2与ax1共享x轴
ax2 = ax1.twinx()
```

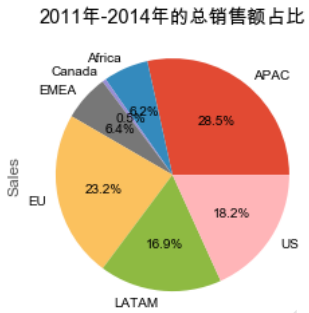
```
ax1.bar(x, y1, color='blue')
ax2.plot(x, y2, marker='*', color='r')
ax1.set_xlabel('年份')
ax1.set_ylabel('销售额')
ax2.set_ylabel('增长率')
ax1.set_title('销售额与增长率')
plt.show()
```

```
Order-year
2011    2.254364e+06
2012    2.665397e+06
2013    3.393116e+06
2014    4.285524e+06
Name: Sales, dtype: float64
0.18232782910632062 0.2730243556885772 0.2630055018020905
18.23% 27.30% 26.30%
              sales_all  sales_rate sales_rate_label
Order-year
2011        2.254364e+06    0.000000            0.00%
2012        2.665397e+06    0.182328           18.23%
2013        3.393116e+06    0.273024           27.30%
2014        4.285524e+06    0.263006           26.30%
```




```
# 各个地区分店的销售额
sales_area = data.groupby(by='Market')['Sales'].sum()
sales_area.plot(kind='pie', autopct="%1.1f%%", title='2011年-2014年的总销售额占比')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f832e7ab550>
```



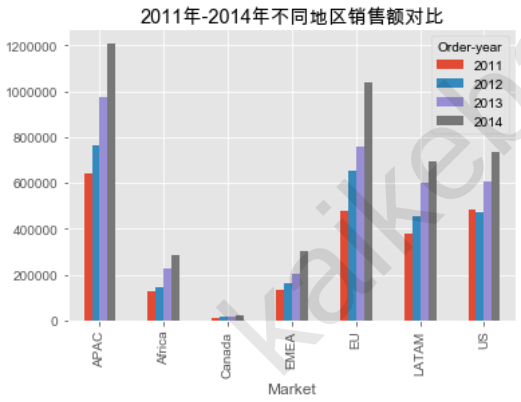
```
# 各地区每一年的销售额
sales_area = data.groupby(by=['Market', 'Order-year'])['Sales'].sum()
# 将分组后的多层索引设置成列数据
sales_area = sales_area.reset_index(level=[0, 1])
# 使用数据透视表重新整理数据
sales_area = pd.pivot_table(sales_area,
                            index='Market',
                            columns='Order-year',
                            values='Sales')
sales_area
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

Order-year	2011	2012	2013	2014
Market				
APAC	639285.99120	762669.96540	974861.9709	1.209074e+06
Africa	127157.21400	144531.09300	229052.2320	2.830364e+05
Canada	8509.11000	16096.80000	19161.1500	2.316111e+04
EMEA	136412.29200	163323.59700	204640.6830	3.015676e+05
EU	477484.33200	651984.77700	756180.6000	1.039217e+06
LATAM	381438.97248	456395.63084	600964.6738	6.957994e+05
US	484076.09010	470395.43300	608254.3660	7.336683e+05

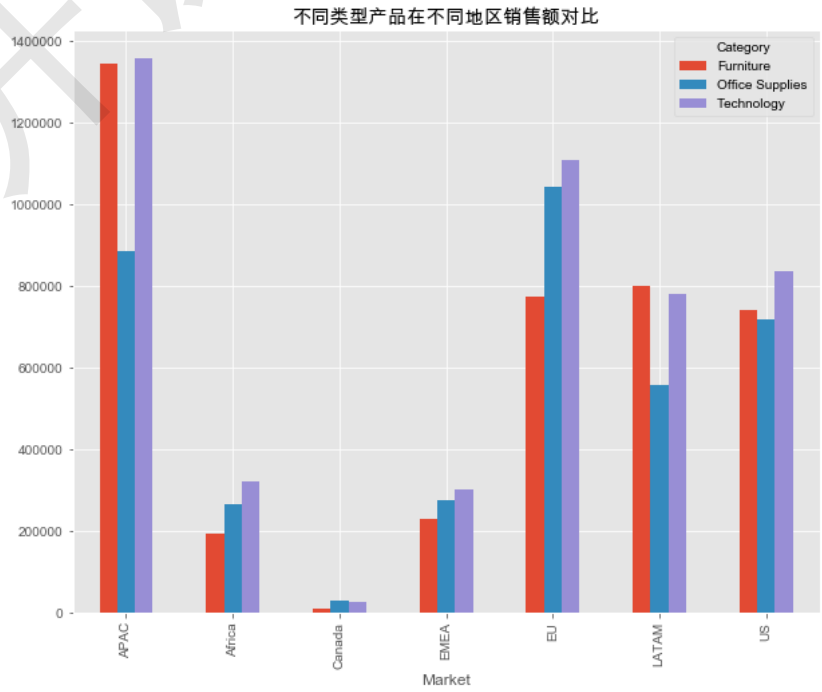
```
# 各地区每一年的销售额
sales_area = data.groupby(by=['Market', 'Order-year'])['Sales'].sum()
# 将分组后的多层索引设置成列数据
sales_area = sales_area.reset_index(level=[0, 1])
# 使用数据透视表重新整理数据
sales_area = pd.pivot_table(sales_area,
                            index='Market',
                            columns='Order-year',
                            values='Sales')
# 绘制图形
sales_area.plot(kind='bar', title='2011年-2014年不同地区销售额对比')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f832ef90310>
```



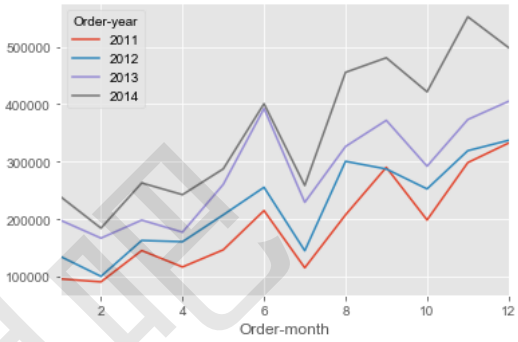
```
# 不同类型产品在不同地区销售额对比
category_sales_area = data.groupby(by=['Market', 'Category'])['Sales'].sum()
category_sales_area
# 将分组后的多层索引设置成列数据
category_sales_area = category_sales_area.reset_index(level=[0, 1])
# 使用数据透视表重新整理数据
category_sales_area = pd.pivot_table(category_sales_area,
                                     index='Market',
                                     columns='Category',
                                     values='Sales')
# 绘制图形
category_sales_area.plot(kind='bar', title='不同类型产品在不同地区销售额对比', figsize=(10, 8))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f832fcded50>
```




```
# 每年每月的销售额对比
year_month = data.groupby(by=['Order-year', 'Order-month'])['Sales'].sum()

# 将索引订单年转为一列数据
sales_year_month = year_month.reset_index(level=[0, 1])
# 利用透视表的确定销售额预览表
sales_year_month = pd.pivot_table(sales_year_month,
                                  index='Order-month',
                                  columns='Order-year',
                                  values='Sales')
# 绘制图形
sales_year_month.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f833087add0>
```




```
# 每年每个月产生新用户的对比
data_customer = data.copy()
data_customer = data_customer.drop_duplicates(subset=['CustomerID'])
new_customer = data_customer.groupby(by=['Order-year', 'Order-month']).size()
new_customer = new_customer.reset_index(level=[0, 1])
customer_year_month = pd.pivot_table(new_customer,
                                     index='Order-month',
                                     columns='Order-year',
                                     values=0,
                                     fill_value=0)

customer_year_month
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

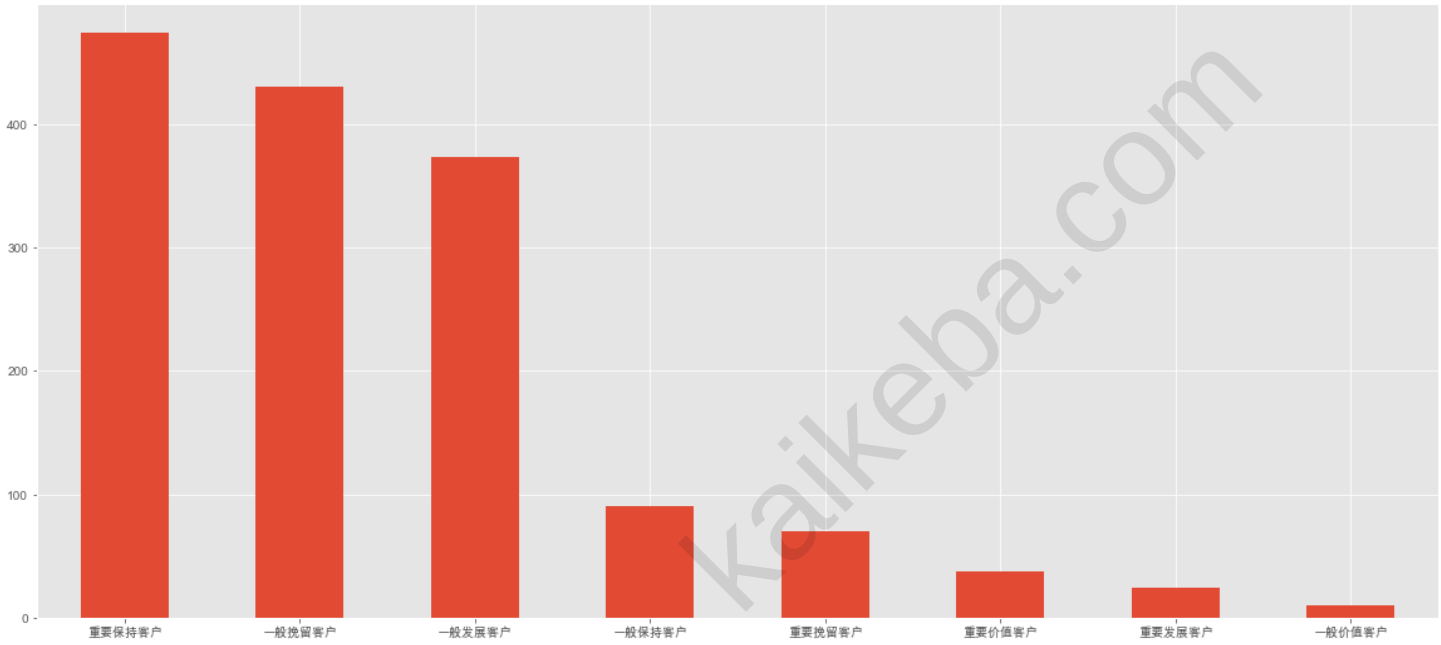
Order-year	2011	2012	2013	2014
Order-month				
1	197	25	5	3
2	139	14	6	3
3	173	18	8	0
4	143	16	5	1
5	114	11	4	0
6	151	28	6	6
7	64	9	4	2
8	106	28	7	0
9	85	22	4	0
10	49	6	2	0
11	47	24	1	0
12	41	9	4	0

```
# 获取2014年数据
data_14 = data[data['Order-year'] == 2014]

# orderData：看到上一次什么时间购买，消费频率
# sales:消费金额
data_14 = data_14[['CustomerID', 'OrderDate', 'Sales']]
print(data_14.shape)
customdf = data_14.copy()

customdf.set_index('CustomerID', drop=True, inplace=True)
```

```
customdf['orders'] = 1
# 2014 8 - 30  2014-8-1
rfmdf = customdf.pivot_table(index=['CustomerID'],
                             values=['OrderDate', 'orders', 'Sales'],
                             aggfunc={
                                 'OrderDate': 'max',
                                 'orders': 'sum',
                                 'Sales': 'sum'
                             })
rfmdf['R'] = (rfmdf.OrderDate.max() - rfmdf.OrderDate).dt.days
rfmdf.rename(columns={'Sales': 'M', 'orders': 'F'}, inplace=True)
rfmdf


def rfm_func(x):
    level = x.apply(lambda x: "1" if x >= 1 else '0')
    label = level.R + level.F + level.M
    d = {
        '111': '重要价值客户',
        '011': '重要保持客户',
        '001': '重要挽留客户',
        '101': '重要发展客户',
        '010': '一般保持客户',
        '110': '一般价值客户',
        '000': '一般挽留客户',
        '100': '一般发展客户'
    }
    result = d[label]
    return result


rfmdf['label'] = rfmdf[['R', 'F',
                        'M']].apply(lambda x: x - x.mean()).apply(rfm_func,
                                                                 axis=1)
rfmdf.groupby('label').count()

rfmdf.label.value_counts().plot.bar(figsize=(20, 9))
plt.xticks(rotation=0)
```

```
(17475, 3)
```

```
(array([0, 1, 2, 3, 4, 5, 6, 7]), <a list of 8 Text xticklabel objects>)
```



```
custom_grade_df = data_14.copy()
custom_grade_df


# 排序函数
def order_sort(group):
    return group.sort_values(by='OrderDate')[-1:]


# 将数据按客户ID分组
data_14_group = data_14.groupby(by='CustomerID', as_index=False)
# 将每个分组对象的数据排序，并取出日期最大的数据
data_max_time = data_14_group.apply(order_sort)
print(data_max_time)
```

```
           CustomerID  OrderDate      Sales
0    50439   AA-10315 2014-12-23    45.9900
1    50605   AA-10375 2014-12-25   444.4200
2    43054   AA-10480 2014-08-28    27.8190
3    49133   AA-10645 2014-12-03    43.2957
4    50872     AA-315 2014-12-29    20.0520
...               ...        ...        ...
1505 50393   YS-21880 2014-12-22  1000.0200
1506 39455   ZC-11910 2014-06-14     7.1730
1507 50843   ZC-21910 2014-12-28   237.3300
1508 50838   ZD-11925 2014-12-28     8.7600
1509 51032   ZD-21925 2014-12-30   216.0000

[1510 rows x 3 columns]
```

```
# 为数据添加F列
data_max_time['F'] = data_14_group.size().values
# 为数据添加M列
data_max_time['M'] = data_14_group.sum()['Sales'].values
print(data_max_time)
```

```
           CustomerID  OrderDate      Sales   F          M
0    50439   AA-10315 2014-12-23    45.9900  17  3889.2065
1    50605   AA-10375 2014-12-25   444.4200  14  1904.5380
2    43054   AA-10480 2014-08-28    27.8190  10  7752.9070
3    49133   AA-10645 2014-12-03    43.2957  19  3539.8788
4    50872     AA-315 2014-12-29    20.0520   3   787.3920
...               ...        ...        ...  ..        ...
1505 50393   YS-21880 2014-12-22  1000.0200  19  7282.4740
1506 39455   ZC-11910 2014-06-14     7.1730   1     7.1730
1507 50843   ZC-21910 2014-12-28   237.3300  27  4922.8390
1508 50838   ZD-11925 2014-12-28     8.7600   8   856.2600
1509 51032   ZD-21925 2014-12-30   216.0000   6  2029.9389

[1510 rows x 5 columns]
```

```
# 确定统计日期
stat_date = pd.to_datetime('2014-12-31')
# 计算最近一次交易时间的间隔
r_data = stat_date - data_max_time['OrderDate']
# 为数据添加R列
data_max_time['R'] = r_data.values
print(data_max_time)
```

```
           CustomerID  OrderDate      Sales   F          M        R
0    50439   AA-10315 2014-12-23    45.9900  17  3889.2065   8 days
1    50605   AA-10375 2014-12-25   444.4200  14  1904.5380   6 days
2    43054   AA-10480 2014-08-28    27.8190  10  7752.9070 125 days
3    49133   AA-10645 2014-12-03    43.2957  19  3539.8788  28 days
4    50872     AA-315 2014-12-29    20.0520   3   787.3920   2 days
...               ...        ...        ...  ..        ...      ...
1505 50393   YS-21880 2014-12-22  1000.0200  19  7282.4740   9 days
1506 39455   ZC-11910 2014-06-14     7.1730   1     7.1730 200 days
1507 50843   ZC-21910 2014-12-28   237.3300  27  4922.8390   3 days
1508 50838   ZD-11925 2014-12-28     8.7600   8   856.2600   3 days
1509 51032   ZD-21925 2014-12-30   216.0000   6  2029.9389   1 days
```

```
[1510 rows x 6 columns]
```

```
section_list_F = [0, 5, 10, 15, 20, 50]
# 根据区间设置评分
grade_F = pd.cut(data_max_time['F'],
                 bins=section_list_F,
                 labels=[1, 2, 3, 4, 5])
# 添加FS评分列
data_max_time['F_S'] = grade_F.values

# 设置M维度的评分
section_list_M = [0, 500, 1000, 5000, 10000, 30000]
# 根据区间设置评分
grade_M = pd.cut(data_max_time['M'],
                 bins=section_list_M,
                 labels=[1, 2, 3, 4, 5])
# 添加MS评分列
data_max_time['M_S'] = grade_M.values

# 设置R维度的评分
import datetime
section_list_R = [
    datetime.timedelta(days=i) for i in [-1, 32, 93, 186, 277, 365]
]
# 根据区间设置评分
grade_R = pd.cut(data_max_time['R'],
                 bins=section_list_R,
                 labels=[5, 4, 3, 2, 1])
# 添加RS评分列
data_max_time['R_S'] = grade_R.values

data_max_time
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

		CustomerID	OrderDate	Sales	F	M	R	F_S	M_S	R_S
0	50439	AA-10315	2014-12-23	45.9900	17	3889.2065	8 days	4	3	5
1	50605	AA-10375	2014-12-25	444.4200	14	1904.5380	6 days	3	3	5
2	43054	AA-10480	2014-08-28	27.8190	10	7752.9070	125 days	2	4	3
3	49133	AA-10645	2014-12-03	43.2957	19	3539.8788	28 days	4	3	5
4	50872	AA-315	2014-12-29	20.0520	3	787.3920	2 days	1	2	5
...	...	...	...	...	...	...	...	...	...	...
1505	50393	YS-21880	2014-12-22	1000.0200	19	7282.4740	9 days	4	4	5
1506	39455	ZC-11910	2014-06-14	7.1730	1	7.1730	200 days	1	1	2
1507	50843	ZC-21910	2014-12-28	237.3300	27	4922.8390	3 days	5	3	5
1508	50838	ZD-11925	2014-12-28	8.7600	8	856.2600	3 days	2	2	5
1509	51032	ZD-21925	2014-12-30	216.0000	6	2029.9389	1 days	2	3	5

1510 rows × 9 columns

```
# 设置F维度高低值
data_max_time['F_S'] = data_max_time['F_S'].values.astype('int')
# 根据评分平均分设置判别高低
grade_avg = data_max_time['F_S'].values.sum() / data_max_time['F_S'].count()
# 将高对应为1，低设置为0
data_F_S = data_max_time['F_S'].where(data_max_time['F_S'] > grade_avg, 0)
```

```
data_max_time['F_high-low'] = data_F_S.where(data_max_time['F_S'] < grade_avg,
                                             1).values

# 设置M维度高低值
data_max_time['M_S'] = data_max_time['M_S'].values.astype('int')
# 根据评分平均分设置判别高低
grade_avg = data_max_time['M_S'].values.sum() / data_max_time['M_S'].count()
# 将高对应为1，低设置为0
data_M_S = data_max_time['M_S'].where(data_max_time['M_S'] > grade_avg, 0)
data_max_time['M_high-low'] = data_M_S.where(data_max_time['M_S'] < grade_avg,
                                             1).values

# 设置R维度高低值
data_max_time['R_S'] = data_max_time['R_S'].values.astype('int')
# 根据评分平均分设置判别高低
grade_avg = data_max_time['R_S'].values.sum() / data_max_time['R_S'].count()
# 将高对应为1，低设置为0
data_R_S = data_max_time['R_S'].where(data_max_time['R_S'] < grade_avg, 0)
data_max_time['R_high-low'] = data_R_S.where(data_max_time['R_S'] > grade_avg,
                                             1).values

# 截取部分列数据
data_rfm = data_max_time[[
    'CustomerID', 'R_high-low', 'F_high-low', 'M_high-low'
]]
data_rfm
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

		CustomerID	R_high-low	F_high-low	M_high-low
0	50439	AA-10315	0	1	1
1	50605	AA-10375	0	1	1
2	43054	AA-10480	1	0	1
3	49133	AA-10645	0	1	1
4	50872	AA-315	0	0	0
...	...	...	...	...	...
1505	50393	YS-21880	0	1	1
1506	39455	ZC-11910	1	0	0
1507	50843	ZC-21910	0	1	1
1508	50838	ZD-11925	0	0	0
1509	51032	ZD-21925	0	0	1

1510 rows × 4 columns

```
custom_grade_df = data_14.copy()
custom_grade_df


# 排序函数
def order_sort(group):
    return group.sort_values(by='OrderDate')[-1:]


# 将数据按客户ID分组
data_14_group = data_14.groupby(by='CustomerID', as_index=False)
# 将每个分组对象的数据排序，并取出日期最大的数据
data_max_time = data_14_group.apply(order_sort)
print(data_max_time)

# 为数据添加F列
```

```
data_max_time['F'] = data_14_group.size().values
# 为数据添加M列
data_max_time['M'] = data_14_group.sum()['Sales'].values
print(data_max_time)

# 确定统计日期
stat_date = pd.to_datetime('2014-12-31')
# 计算最近一次交易时间的间隔
r_data = stat_date - data_max_time['OrderDate']
# 为数据添加R列
data_max_time['R'] = r_data.values
print(data_max_time)

section_list_F = [0, 5, 10, 15, 20, 50]
# 根据区间设置评分
grade_F = pd.cut(data_max_time['F'],
                 bins=section_list_F,
                 labels=[1, 2, 3, 4, 5])
# 添加FS评分列
data_max_time['F_S'] = grade_F.values

# 设置M维度的评分
section_list_M = [0, 500, 1000, 5000, 10000, 30000]
# 根据区间设置评分
grade_M = pd.cut(data_max_time['M'],
                 bins=section_list_M,
                 labels=[1, 2, 3, 4, 5])
# 添加MS评分列
data_max_time['M_S'] = grade_M.values

# 设置R维度的评分
import datetime
section_list_R = [
    datetime.timedelta(days=i) for i in [-1, 32, 93, 186, 277, 365]
]
# 根据区间设置评分
grade_R = pd.cut(data_max_time['R'],
                 bins=section_list_R,
                 labels=[5, 4, 3, 2, 1])
# 添加RS评分列
data_max_time['R_S'] = grade_R.values

# 设置F维度高低值
data_max_time['F_S'] = data_max_time['F_S'].values.astype('int')
# 根据评分平均分设置判别高低
grade_avg = data_max_time['F_S'].values.sum() / data_max_time['F_S'].count()
# 将高对应为1，低设置为0
data_F_S = data_max_time['F_S'].where(data_max_time['F_S'] > grade_avg, 0)
data_max_time['F_high-low'] = data_F_S.where(data_max_time['F_S'] < grade_avg,
                                             1).values

# 设置M维度高低值
data_max_time['M_S'] = data_max_time['M_S'].values.astype('int')
# 根据评分平均分设置判别高低
grade_avg = data_max_time['M_S'].values.sum() / data_max_time['M_S'].count()
# 将高对应为1，低设置为0
data_M_S = data_max_time['M_S'].where(data_max_time['M_S'] > grade_avg, 0)
data_max_time['M_high-low'] = data_M_S.where(data_max_time['M_S'] < grade_avg,
                                             1).values

# 设置R维度高低值
data_max_time['R_S'] = data_max_time['R_S'].values.astype('int')
# 根据评分平均分设置判别高低
grade_avg = data_max_time['R_S'].values.sum() / data_max_time['R_S'].count()
# 将高对应为1，低设置为0
data_R_S = data_max_time['R_S'].where(data_max_time['R_S'] < grade_avg, 0)
data_max_time['R_high-low'] = data_R_S.where(data_max_time['R_S'] > grade_avg,
                                             1).values

# 截取部分列数据
data_rfm = data_max_time[[
    'CustomerID', 'R_high-low', 'F_high-low', 'M_high-low'
]]


def get_sum_value(series):
    return ''.join([str(i) for i in series.values.tolist()[1:]])
```

```
# 添加RFM字符串列
data_rfm['data_rfm'] = data_rfm.apply(get_sum_value, axis=1)
dic = {
    '111': '重要价值客户',
    '011': '重要保持客户',
    '001': '重要挽留客户',
    '101': '重要发展客户',
    '010': '一般保持客户',
    '110': '一般价值客户',
    '000': '一般挽留客户',
    '100': '一般发展客户'
}
# RFM字符串数据映射成对应类型文字
data_rfm['data_rfm'] = data_rfm['data_rfm'].map(dic)
# print(data_rfm)

size = data_rfm.groupby(by='data_rfm').size()
size = size.to_frame()
size['rfm_pct'] = [
    "%.2f%%" % (i / sum(size.values) * 100) for i in size.values
]
size
```

```
           CustomerID  OrderDate      Sales
0    50439   AA-10315 2014-12-23    45.9900
1    50605   AA-10375 2014-12-25   444.4200
2    43054   AA-10480 2014-08-28    27.8190
3    49133   AA-10645 2014-12-03    43.2957
4    50872     AA-315 2014-12-29    20.0520
...              ...        ...        ...
1505 50393   YS-21880 2014-12-22  1000.0200
1506 39455   ZC-11910 2014-06-14     7.1730
1507 50843   ZC-21910 2014-12-28   237.3300
1508 50838   ZD-11925 2014-12-28     8.7600
1509 51032   ZD-21925 2014-12-30   216.0000

[1510 rows x 3 columns]
           CustomerID  OrderDate      Sales   F          M
0    50439   AA-10315 2014-12-23    45.9900  17  3889.2065
1    50605   AA-10375 2014-12-25   444.4200  14  1904.5380
2    43054   AA-10480 2014-08-28    27.8190  10  7752.9070
3    49133   AA-10645 2014-12-03    43.2957  19  3539.8788
4    50872     AA-315 2014-12-29    20.0520   3   787.3920
...              ...        ...        ...  ..        ...
1505 50393   YS-21880 2014-12-22  1000.0200  19  7282.4740
1506 39455   ZC-11910 2014-06-14     7.1730   1     7.1730
1507 50843   ZC-21910 2014-12-28   237.3300  27  4922.8390
1508 50838   ZD-11925 2014-12-28     8.7600   8   856.2600
1509 51032   ZD-21925 2014-12-30   216.0000   6  2029.9389

[1510 rows x 5 columns]
           CustomerID  OrderDate      Sales   F          M        R
0    50439   AA-10315 2014-12-23    45.9900  17  3889.2065   8 days
1    50605   AA-10375 2014-12-25   444.4200  14  1904.5380   6 days
2    43054   AA-10480 2014-08-28    27.8190  10  7752.9070 125 days
3    49133   AA-10645 2014-12-03    43.2957  19  3539.8788  28 days
4    50872     AA-315 2014-12-29    20.0520   3   787.3920   2 days
...              ...        ...        ...  ..        ...      ...
1505 50393   YS-21880 2014-12-22  1000.0200  19  7282.4740   9 days
1506 39455   ZC-11910 2014-06-14     7.1730   1     7.1730 200 days
1507 50843   ZC-21910 2014-12-28   237.3300  27  4922.8390   3 days
1508 50838   ZD-11925 2014-12-28     8.7600   8   856.2600   3 days
1509 51032   ZD-21925 2014-12-30   216.0000   6  2029.9389   1 days

[1510 rows x 6 columns]
```

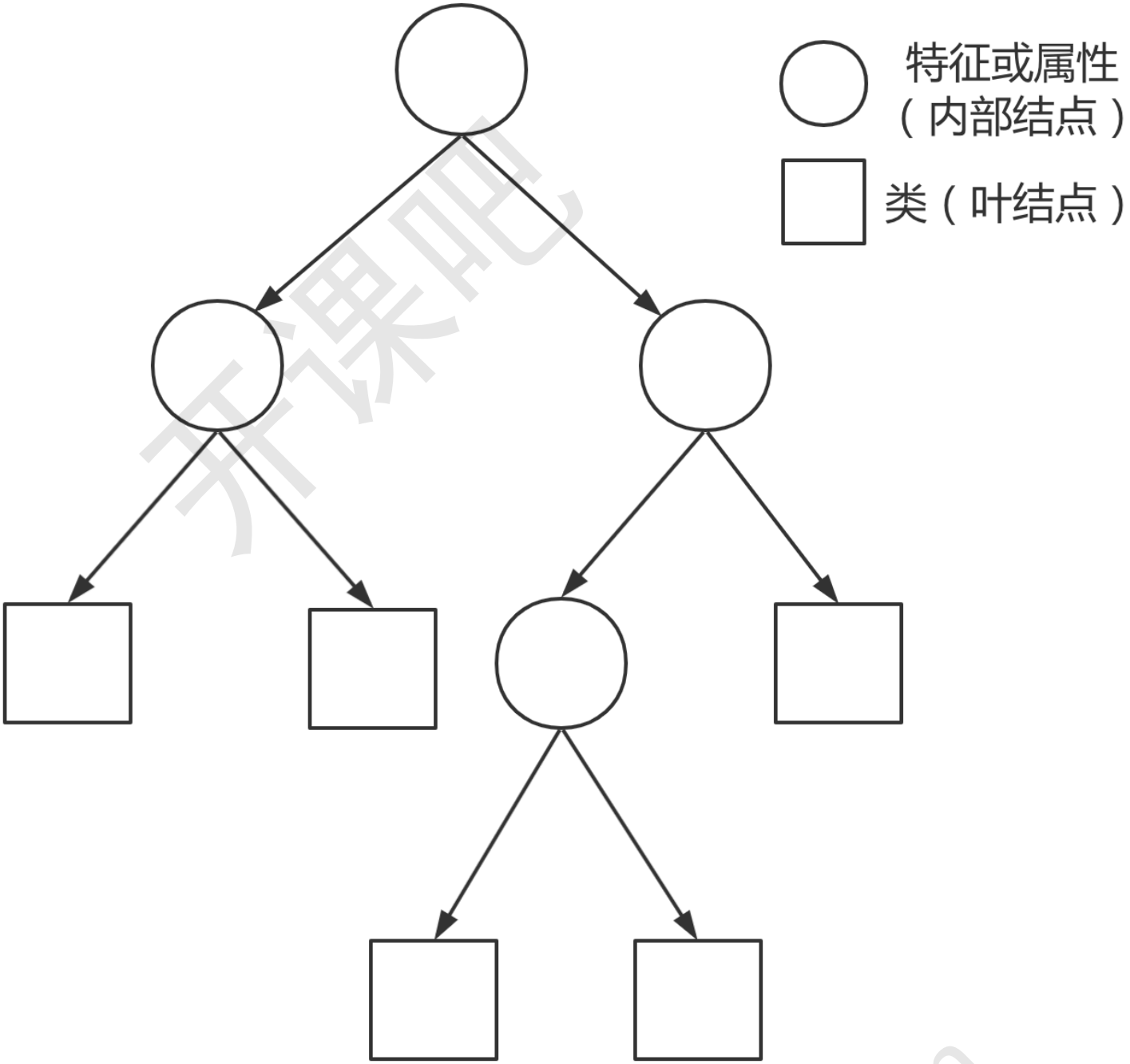
```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	0	rfm_pct
data_rfm		
一般价值客户	5	0.33%
一般保持客户	8	0.53%
一般发展客户	402	26.62%
一般挽留客户	145	9.60%
重要价值客户	229	15.17%
重要保持客户	471	31.19%
重要发展客户	139	9.21%
重要挽留客户	111	7.35%

4.4 决策树回顾

决策树的定义：分类决策树模型是一种描述对实例进行分类的树形结构，决策树由节点和有向边组成，节点有两种类型：内部节点和叶子结点，内部结点表示一个特征或属性，叶结点表示一个类，一般的一棵决策树包含一个根节点（最顶部的内部结点）若干个内部结点和若干个叶子结点。

简单的用if-then来说一下，决策树可以看作是一个if-then规则的集合，if用来表示属性（特征），then用来表示分类结果，简单的结构图如下所示：



4.4.1 决策树的划分

4.4.2 信息增益

在构造决策树之前，根据树形结构我们考虑在进行分类的时候哪个特征起到了更重要的作用，这样的特征我们是不是应该把它放在更重要的位置呢，又要怎样去确定一个特征是否重要呢？这个时候信息增益就起到了很大的作用。

先从熵说起。

熵

熵通常出现在信息论和概率统计中，通常用来度量随机变量的不确定性。

设X是一个取有限个值的离散型随机变量，其概率分布为：

$P(X=x_i)=P_i, i=1,2,3,\ldots,n$

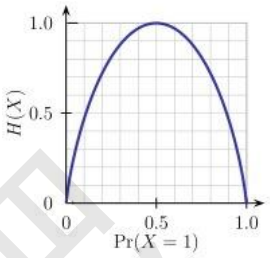
则随机变量X的熵为：

$H(X)=-\sum_{i=1}^{n} p_i log(p_i)$ （通常式子中的log会以2为底或者以e为底）

根据上式我们可以看到式子中并没有涉及到X的取值，而是涉及到了X的分布，所以我们可以认为熵只和X的分布有关，是和X的取值无关的，所以我们也可以把熵的式子定义成：

$H(P)=-\sum_{i=1}^{n} p_i log(p_i)$

对于伯努利分布来说，H(p)随概率P的变化如下图所示：



根据图像可以知道，当P=0和P=1时，H(p)=0，随机变量完全没有不确定性，而当P=0.5时，H(p)=1,随机变量的不确定性最大。举个例子来说就是当我们抛一个两面都一样的硬币时，可以得到的结果是确定的，而抛正常的硬币时，是完全随机的，也就是不确定性很大，此时熵也是最大的。

条件熵

条件熵H(Y|X)表示的是在已知随机变量X的情况下，随机变量Y的不确定性，条件熵的定义如下：

$H(Y|X) = \sum_{i=1}^{n} p_i H(Y|X = x_i),$

$pi = p(X = x_i),$

$i = 1, 2, 3, \ldots, n$

信息增益

信息增益表示的是得知X的信息而使得类Y的信息不确定性减少的程度。

定义：特征A对训练数据集D的信息增益Gain(D,A)，定义为集合D的经验熵H(D)与特征A给定条件下D的经验条件熵H(D|A)之差，即：

$Gain(D, A) = H(D) - H(D|A)$

通常情况下我们把熵和条件熵的差称作为互信息，决策树学习中的信息增益等价于训练数据集中类与特征的互信息。

对于数据集D而言，信息增益依赖于特征，不同的特征往往具有不同的信息增益，信息增益大的特征具有更强的分类能力。

当我们进行特征选择的时候对训练数据集D，计算其每个特征的信息增益，并比较他们的大小，选择信息增益最大的特征。

4.4.3 决策树生成（ID3算法）

ID3算法就是在信息增益的基础上使用信息增益来选择特征，递归的来构建决策树。

主要方法

从根节点开始，对节点计算所有可能特征的信息增益，选择信息增益最大的特征作为节点的特征，并且由该特征取不同的值来建立子节点，再对子节点递归调用上述方法来构建决策树，知道所有的信息增益都变得很小或者没有特征可以选择的时候，得到最终的决策树。

ID3示例

光看概念和步骤难免会有些抽象，让我们通过一个例子来看一下ID3的具体计算和树的生成过程。

下表是西瓜书中的西瓜数据集2.0，该数据集中包含17个训练样例，用来学习预测一个没切开的瓜是不是好瓜。很显然这是一个二分类问题。

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否

在决策树开始构建之前，我们先看一下正例和反例所占的比例：

正例：$p_1 = \frac{8}{17}$

反例：$p_2 = \frac{9}{17}$

根结点的信息熵：$H(D) = -\sum_{i=1}^{2} p_i log(p_i) = -(\frac{8}{17} log_2 \frac{8}{17} + \frac{9}{17} log_2 \frac{9}{17}) = 0.998$

然后我们需要计算出当前属性集合{色泽，根蒂，敲声，纹理，脐部，触感}中每一个属性的信息增益。

以属性"色泽"为例，他有三个取值：{青绿、乌黑、浅白}，若使用该属性对D进行划分，则可得到三个子集，分别记为D1（色泽=青绿），D2（色泽=乌黑），D3（色泽=浅白）。根据上表我们可以得到如下的信息：

数据集D1：{1、4、6、10、13、17}，正例p1=3/6，反例p2=3/6

数据集D2：{2、3、7、8、9、15}，正例p1=4/6，反例p2=2/6

数据集D3：{5、11、12、14、16}，正例p1=1/5，反例p2=4/5

我们可以计算他们对应的信息增益如下：

$H(D_1) = -(\frac{3}{6} log_2 \frac{3}{6} + \frac{3}{6} log_2 \frac{3}{6}) = 1.000$

$H(D_2) = -(\frac{4}{6} log_2 \frac{4}{6} + \frac{2}{6} log_2 \frac{2}{6}) = 0.918$

$H(D_3) = -(\frac{1}{5} log_2 \frac{1}{5} + \frac{4}{5} log_2 \frac{4}{5}) = 0.722$

对应的对于属性"色泽"的信息增益有：

$Gain(D, \text{色泽}) = H(D) - \sum_{i=1}^{3} H(D|\text{色泽}_i)$

$Gain(D, \text{色泽}) = 0.998 - (\frac{6}{17} * 1.000 + \frac{6}{17} * 0.918 + \frac{5}{17} * 0.722) = 0.109$

类似的我们可以算出其他属性的信息增益如下：

$Gain(D, 根蒂) = 0.143$

$Gain(D, 敲声) = 0.141$

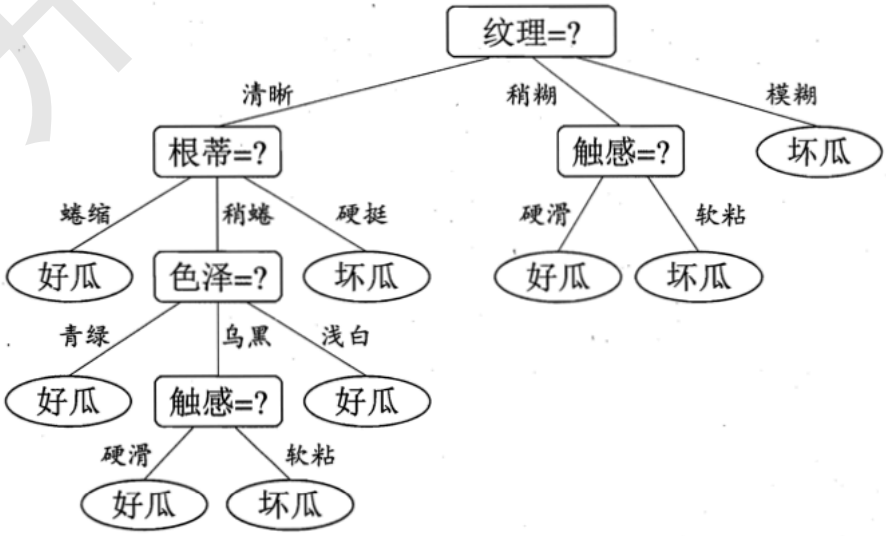
$Gain(D, 纹理) = 0.381$

$Gain(D, 脐部) = 0.289$

$Gain(D, 触感) = 0.006$

很显然，纹理的信息增益最大，那么我们就选择它为划分属性，得到的结果如下：

对于每一个结点，我们又可以计算其余特征的信息增益进行划分，如此递归可以得到最终的决策树如下：



4.4.4 信息增益率

根据信息增益的概念我们会发现，随着数据量的增大，信息增益会偏向取值较多的特征，使用信息增益率可以对这一问题进行校正。

定义：特征A对训练数据集D的信息增益比GainRatio(D,A)定义为其信息增益Gain(D,A)与训练数据集D的经验熵H(D)之比，即：

$$GainRatio(D, A) = \frac{Gain(D,A)}{H(D)}$$

这里的Gain也就是我们的信息增益了，而H(D)也就是对应的熵了。

基于信息增益率我们有对应的算法叫做C4.5算法，需要注意的是，增益率可以对取值数目较少的属性，因此C4.5算法并不是直接选择增益率最大的候选划分属性，而是使用了一个启发式（先从候选划分属性中找出信息增益高于平均水平的属性，再从中选择信息增益率最高的）

4.4.5 基尼指数（Gini）

从之前的内容我们了解到ID3使用信息增益来选择特征，C4.5使用信息增益率来选择特征，同样对于我们的CART需要使用基尼系数来进行特征选择。

基尼系数代表了了模型的不纯度，基尼系数越小，不纯度越低，特征越好，这和信息增益比是相反的。先来看一下基尼系数的定义：

$$Gini(D) = \sum_{k=1}^{k} P_k(1 - P_k) = 1 - \sum_{k=1}^{k} (\frac{|C_k|}{|D|})^2$$

（C_k是D中属于第K类的样本子集，K是类的个数）

CART在每一次迭代中选择基尼指数最小的特征及其对应的切分点进行分类，与ID3、C4.5不同的是，CART是一颗二叉树，采用二元切割法，每一步将数据按特征A的取值切分成两份，分别进入左右子树，特征A的基尼指数为：

$$Gini(D|A) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} Gini(Di)$$

4.4.6 决策树的优化（剪枝）

当我们把所有的特征都放入决策树中的时候，树的模型会显得很大，同时也大概率的会造成过拟合现象的产生，为了避免该现象的发生，我们需要对决策树进行**剪枝**。

决策树的剪枝通常有两种方法：预剪枝和后剪枝。

预剪枝

预剪枝的核心思想是在树中节点进行扩展之前，先计算当前的划分是否能够带来模型泛化能力的提升，如果不能则不再继续生长子树。

预剪枝对于何时停止决策树的生长有如下的几种方法：

（1）当树达到一定深度的时候，停止树的生长。

（2）当达到前节点的样本数量小于某个阈值的时候，停止树的生长。

（3）计算每次分裂对测试集的准确度的提升，当小于某个阈值的守，不在继续扩展。

后剪枝

后剪枝的思想是让算法生成一颗完全生长的决策树，然后从最上层计算是否剪枝，剪枝过程将子树删除，用一个叶子节点代替，该节点的类别同样按照多数投票的原则进行判断。

4.4.7 决策树算法优缺点

优点：

1. 简单直观，生成的决策树很直观。
2. 基本不需要预处理，不需要提前归一化和处理缺失值。
3. 使用决策树预测的代价是O(log2m)。m为样本数。
4. 既可以处理离散值也可以处理连续值。很多算法只是专注于离散值或者连续值。
5. 可以处理多维度输出的分类问题。
6. 相比于神经网络之类的黑盒分类模型，决策树在逻辑上可以很好解释。
7. 可以交叉验证的剪枝来选择模型，从而提高泛化能力。
8. 对于异常点的容错能力好，健壮性高。

缺点：

1. 决策树算法非常容易过拟合，导致泛化能力不强。可以通过设置节点最少样本数量和限制决策树深度来改进。
2. 决策树会因为样本发生一点的改动，导致树结构的剧烈改变。这个可以通过集成学习之类的方法解决。
3. 寻找最优的决策树是一个NP难题，我们一般是通过启发式方法，容易陷入局部最优。可以通过集成学习的方法来改善。
4. 有些比较复杂的关系，决策树很难学习，比如异或。这个就没有办法了，一般这种关系可以换神经网络分类方法来解决。
5. 如果某些特征的样本比例过大，生成决策树容易偏向于这些特征。这个可以通过调节样本权重来改善。

4.5 决策树-用户分层

```
data_rfm
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

		CustomerID	R_high-low	F_high-low	M_high-low	data_rfm
0	50439	AA-10315	0	1	1	重要保持客户
1	50605	AA-10375	0	1	1	重要保持客户
2	43054	AA-10480	1	0	1	重要发展客户
3	49133	AA-10645	0	1	1	重要保持客户
4	50872	AA-315	0	0	0	一般挽留客户
...	...	...	...	...	...	...
1505	50393	YS-21880	0	1	1	重要保持客户
1506	39455	ZC-11910	1	0	0	一般发展客户
1507	50843	ZC-21910	0	1	1	重要保持客户
1508	50838	ZD-11925	0	0	0	一般挽留客户
1509	51032	ZD-21925	0	0	1	重要挽留客户

1510 rows × 5 columns

```
pd.DataFrame(data_rfm['data_rfm'])
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

		data_rfm
0	50439	重要保持客户
1	50605	重要保持客户
2	43054	重要发展客户
3	49133	重要保持客户
4	50872	一般挽留客户
...	...	...
1505	50393	重要保持客户
1506	39455	一般发展客户
1507	50843	重要保持客户
1508	50838	一般挽留客户
1509	51032	重要挽留客户

1510 rows × 1 columns

```
data_max_time['data_rfm'] = data_rfm['data_rfm']
data_max_time
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

		CustomerID	OrderDate	Sales	F	M	R	F_S	M_S	R_S	F_high-low	M_high-low	R_high-low	data_rfm
0	50439	AA-10315	2014-12-23	45.9900	17	3889.2065	8 days	4	3	5	1	1	0	重要保持客户
1	50605	AA-10375	2014-12-25	444.4200	14	1904.5380	6 days	3	3	5	1	1	0	重要保持客户
2	43054	AA-10480	2014-08-28	27.8190	10	7752.9070	125 days	2	4	3	0	1	1	重要发展客户
3	49133	AA-10645	2014-12-03	43.2957	19	3539.8788	28 days	4	3	5	1	1	0	重要保持客户
4	50872	AA-315	2014-12-29	20.0520	3	787.3920	2 days	1	2	5	0	0	0	一般挽留客户
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1505	50393	YS-21880	2014-12-22	1000.0200	19	7282.4740	9 days	4	4	5	1	1	0	重要保持客户
1506	39455	ZC-11910	2014-06-14	7.1730	1	7.1730	200 days	1	1	2	0	0	1	一般发展客户
1507	50843	ZC-21910	2014-12-28	237.3300	27	4922.8390	3 days	5	3	5	1	1	0	重要保持客户
1508	50838	ZD-11925	2014-12-28	8.7600	8	856.2600	3 days	2	2	5	0	0	0	一般挽留客户
1509	51032	ZD-21925	2014-12-30	216.0000	6	2029.9389	1 days	2	3	5	0	1	0	重要挽留客户

1510 rows × 13 columns

```
new_df = data_max_time[['F','M','R','data_rfm']]
new_df
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

		F	M	R	data_rfm
0	50439	17	3889.2065	8 days	重要保持客户
1	50605	14	1904.5380	6 days	重要保持客户
2	43054	10	7752.9070	125 days	重要发展客户
3	49133	19	3539.8788	28 days	重要保持客户
4	50872	3	787.3920	2 days	一般挽留客户
...	...	...	...	...	...
1505	50393	19	7282.4740	9 days	重要保持客户
1506	39455	1	7.1730	200 days	一般发展客户
1507	50843	27	4922.8390	3 days	重要保持客户
1508	50838	8	856.2600	3 days	一般挽留客户
1509	51032	6	2029.9389	1 days	重要挽留客户

1510 rows × 4 columns

```
new_df.reset_index(drop=True,inplace=True)
new_df
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	F	M	R	data_rfm
0	17	3889.2065	8 days	重要保持客户
1	14	1904.5380	6 days	重要保持客户
2	10	7752.9070	125 days	重要发展客户
3	19	3539.8788	28 days	重要保持客户
4	3	787.3920	2 days	一般挽留客户
...	...	...	...	...
1505	19	7282.4740	9 days	重要保持客户
1506	1	7.1730	200 days	一般发展客户
1507	27	4922.8390	3 days	重要保持客户
1508	8	856.2600	3 days	一般挽留客户
1509	6	2029.9389	1 days	重要挽留客户

1510 rows × 4 columns

```
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1510 entries, 0 to 1509
Data columns (total 4 columns):
F           1510 non-null int64
M           1510 non-null float64
R           1510 non-null timedelta64[ns]
data_rfm    1510 non-null object
dtypes: float64(1), int64(1), object(1), timedelta64[ns](1)
memory usage: 47.3+ KB
```

```
new_df['R'].loc[0].days
```

```
8
```

```
new_df['R'] = new_df['R'].apply(lambda x: x.days)
```

```
new_df['R'].values
```

```
array([  8,   6, 125, ...,   3,   3,   1])
```

```
# 获取数据
from sklearn.tree import DecisionTreeClassifier

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

```
from sklearn.model_selection import train_test_split
data_x = new_df.drop(['data_rfm'], axis=1)
data_y = new_df['data_rfm']
train_x, test_x, train_y, test_y = train_test_split(data_x,
                                                    data_y,
                                                    test_size=0.2,
                                                    random_state=8)

train_x
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	F	M	R
300	15	5986.2625	15
1344	1	21.1200	169
529	8	5183.5200	22
461	3	670.4760	111
955	1	28.9800	48
...	...	...	...
986	8	1165.7100	40
133	9	972.1700	9
361	25	8351.8480	7
1364	8	830.7000	95
451	15	890.7180	33

1208 rows × 3 columns

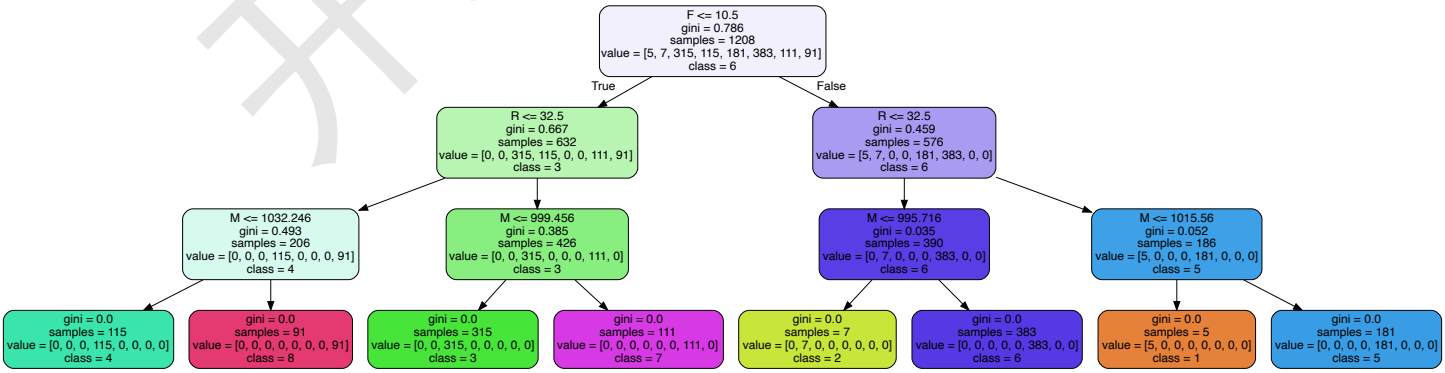
```
train_x.shape
```

```
(1208, 3)
```

```
# 使用决策树算法
clf = DecisionTreeClassifier(criterion='gini',
                             max_depth=3,
                             min_samples_leaf=5,
                             min_samples_split=2,
                             random_state=0)
clf.fit(train_x, train_y)
score = clf.score(test_x, test_y)
score
```

```
0.9966887417218543
```

```
import os

plt.rcParams['font.sans-serif'] = ['SimHei']  #显示中文标签
plt.rcParams["font.family"] = 'Arial Unicode MS'  # mac
plt.rcParams['axes.unicode_minus'] = False  #用来正常显示负号
from sklearn import tree
import graphviz
#颜色越浅，不纯度越高
dot = tree.export_graphviz(clf,# DT
                           feature_names=train_x.columns,# RFM
                           class_names=['1', '2','3','4','5','6','7','8'],#
                           filled=True,
                           rounded=True)

graph = graphviz.Source(dot)
graph
```



五、总结

- 熟悉RFM的基本原理
- 制作RFM用户分层项目
- 制作RFM决策树用户分层

六、作业

- 总结用户分层的思想和应用方法