第六部分 数据分析综合实例（泰坦尼克号乘客数据分析）

6-1 数据读入和查看

```
# 读入训练数据
import pandas as pd
import numpy as np
data = pd.read_csv("data/titan_train.csv")
data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

特征含义：

- PassengerId：乘客编号（无意义）
- Survived：是否存活（1-存活，0-未存活），目标特征
- Pclass：船舱等级（1、2、3等舱）
- SibSp：堂兄弟姐妹个数
- Parch：直系亲属个数
- Embarked：登船港口

```
# 看一看数据的基本信息
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

我们发现：

- 乘客总数（记录数）：891
- 特征总数：12
- 缺失：年龄、船舱编号、登船港口

```
# 看一看数据的基本统计值
data.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

我们可以看出：

- 平均年龄29.7岁，说明乘客青壮年居多
- 存活率38.4%
- 2、3等舱乘客比一等舱要多很多

6-2 缺失值处理

```
# 读入训练数据
import pandas as pd
import numpy as np
data = pd.read_csv("data/titan_train.csv")
data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
# 对年龄缺失值的处理，采用平均值填充
import numpy as np
aver_age = np.round(np.mean(data.Age),1)
data.Age[data.Age.isnull()] = aver_age
print("平均年龄：", data["Age"].mean())
data.info()
```

```
平均年龄： 29.69929292929302
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
```

```
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
<ipython-input-5-8b884bd9afac>:4: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data.Age[data.Age.isnull()] = aver_age
```

处理后可以看出，平均年龄不变，年龄特征已经没有缺失值了。

```
# 对于船舱编号的处理，有船舱编号的设为Yes，无No
data.loc[data.Cabin.notnull(),"Cabin"] = "Yes"
data.loc[data.Cabin.isnull(),"Cabin"] = "No"
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        891 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	No	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	Yes	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	No	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	Yes	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	No	S

```
# 对于登船港口的处理，我们采用最频繁的值填充
data.Embarked.value_counts()
```

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

```
# 我们用最多的“S”来填充缺失的登船港口
data.loc[data.Embarked.isnull(),"Embarked"] = "S"
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        891 non-null    object 
 11  Embarked     891 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

经过处理之后，发现已经没有缺失值，简单的数据预处理告一段落。

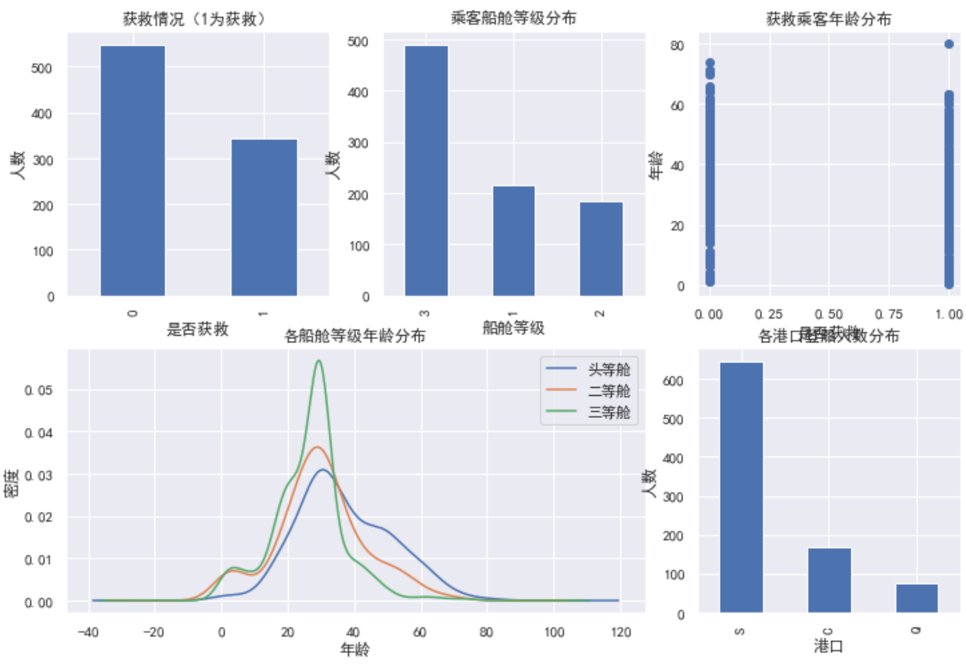
```
# 保存数据
data.to_csv("data/titan.csv", index=False)
```

6-3 数据特征分析

```
# 读入数据
data = pd.read_csv("data/titan.csv")
data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	No	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	Yes	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	No	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	Yes	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	No	S

```python
# 引入绘图库并设置相关参数（中文处理）
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
plt.rcParams["font.sans-serif"] = "SimHei"
plt.rcParams["axes.unicode_minus"] = False

# 设置图形大小
fig = plt.figure(figsize=(12,8))
fig.set(alpha=0.2)

# 设置子图，绘制获救情况的条形图
plt.subplot2grid((2,3),(0,0))
data.Survived.value_counts().plot(kind="bar")
plt.title("获救情况（1为获救）")
plt.xlabel("是否获救")
plt.ylabel("人数")

# 绘制乘客船舱等级分布
plt.subplot2grid((2,3),(0,1))
data.Pclass.value_counts().plot(kind="bar")
plt.title("乘客船舱等级分布")
plt.xlabel("船舱等级")
plt.ylabel("人数")

# 绘制获救和年龄之间的关系的散点图
plt.subplot2grid((2,3),(0,2))
plt.scatter(data.Survived, data.Age)
plt.title("获救乘客年龄分布")
plt.xlabel("是否获救")
plt.ylabel("年龄")
plt.grid(b=True,which="major", axis="y")

# 绘制各船舱等级的年龄分布
plt.subplot2grid((2,3),(1,0),colspan=2)
data.Age[data.Pclass==1].plot(kind="kde")
data.Age[data.Pclass==2].plot(kind="kde")
data.Age[data.Pclass==3].plot(kind="kde")
plt.xlabel("年龄")
plt.ylabel("密度")
plt.title("各船舱等级年龄分布")
plt.legend(["头等舱","二等舱","三等舱"])

# 绘制各港口登船人数分布
plt.subplot2grid((2,3),(1,2))
data.Embarked.value_counts().plot(kind="bar")
plt.title("各港口登船人数分布")
plt.xlabel("港口")
plt.ylabel("人数")
plt.show()
```



从上面的分析可以看出：

- 获救人数300多点，不到半数；
- 3等舱乘客非常多，超过半数；
- 遇难和获救乘客年龄分布非常广，没有特别的规律，但是60岁以上基本全部遇难；
- 2、3等舱主要是20-30岁之间的乘客，头等舱主要40岁以上；
- S港口登船人数最多（南安普顿）。

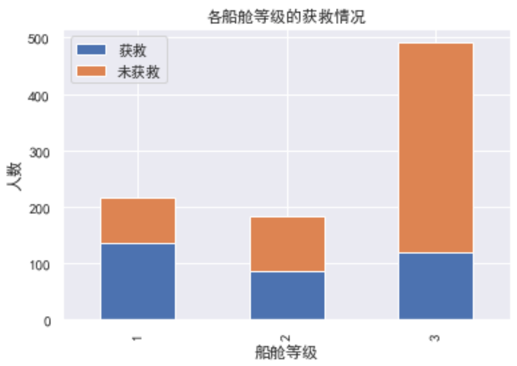
6-4 各特征和是否存活的关系

```
# 读入数据
data = pd.read_csv("data/titan.csv")
data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	No	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	Yes	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	No	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	Yes	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	No	S

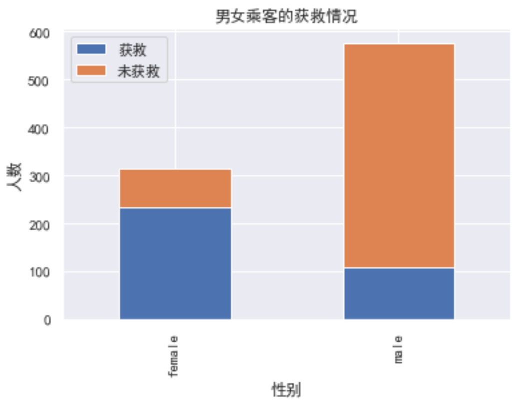
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
plt.rcParams["font.sans-serif"] = "SimHei"
plt.rcParams["axes.unicode_minus"] = False
```

```
# 船舱等级和获救之间的关系分析
# 绘图库引入和设置（略）
# 计算各等级获救和遇难的人数
s_0 = data.Pclass[data.Survived==0].value_counts()
s_1 = data.Pclass[data.Survived==1].value_counts()
# 创建一个数据框
df = pd.DataFrame({"获救":s_1, "未获救":s_0})
# 绘制层叠条形图（使用pandas绘图）
df.plot(kind="bar", stacked=True)
plt.title("各船舱等级的获救情况")
plt.xlabel("船舱等级")
plt.ylabel("人数")
plt.show()
```



可以看出：头等舱获救比例非常高，3等舱获救比例非常低，财富真能买到生命？

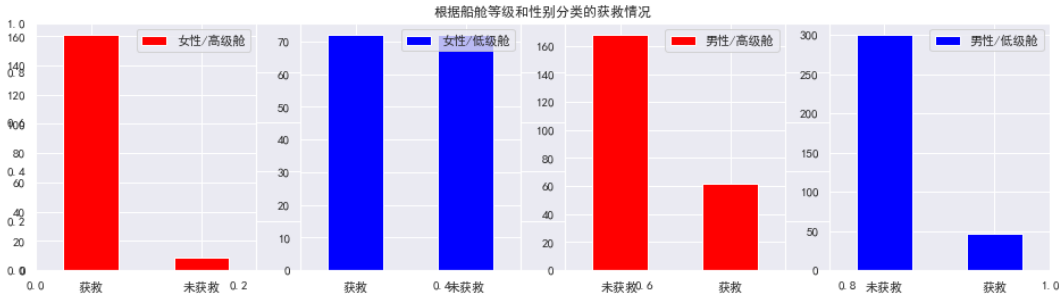
```
# 计算男性和女性获救和遇难人数
s_0 = data.Sex[data.Survived==0].value_counts()
s_1 = data.Sex[data.Survived==1].value_counts()
df = pd.DataFrame({"获救":s_1, "未获救":s_0})
# 绘制层叠条形图（使用pandas绘图）
df.plot(kind="bar", stacked=True)
plt.title("男女乘客的获救情况")
plt.xlabel("性别")
plt.ylabel("人数")
plt.show()
```



女性获救比例远远高于男性，说明女士优先的绅士文化根深蒂固。

```
# 船舱等级和性别对于获救的综合分析
fig = plt.figure(figsize=(16,4))
fig.set(alpha=0.6)
plt.title("根据船舱等级和性别分类的获救情况")
# 女性1、2等舱
ax1 = fig.add_subplot(141)
data.Survived[data.Sex=="female"][data.Pclass!=3].value_counts().plot(kind="bar",color="red")
ax1.set_xticklabels(["获救","未获救"], rotation=0)
ax1.legend(["女性/高级舱"], loc="best")
# 女性3等舱
ax2 = fig.add_subplot(142)
data.Survived[data.Sex=="female"][data.Pclass==3].value_counts().plot(kind="bar",color="blue")
ax2.set_xticklabels(["获救","未获救"], rotation=0)
ax2.legend(["女性/低级舱"], loc="best")
# 男性1、2等舱
ax3 = fig.add_subplot(143)
data.Survived[data.Sex=="male"][data.Pclass!=3].value_counts().plot(kind="bar",color="red")
ax3.set_xticklabels(["未获救","获救"], rotation=0)
```

```
ax3.legend(["男性/高级舱"], loc="best")
# 男性3等舱
ax4 = fig.add_subplot(144)
data.Survived[data.Sex=="male"][data.Pclass==3].value_counts().plot(kind="bar",color="blue")
ax4.set_xticklabels(["未获救","获救"], rotation=0)
ax4.legend(["男性/低级舱"], loc="best")

plt.show()
```

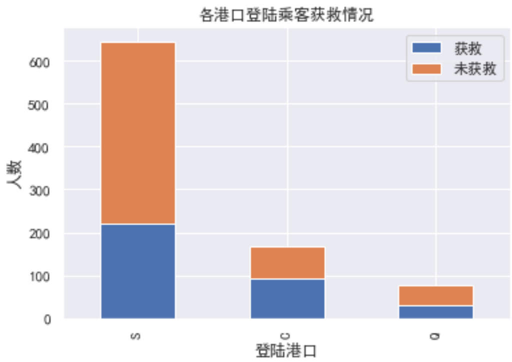


几点结论:

- 高级舱女性几乎全部获救
- 高级舱男性没有女性的运气，但高级舱比低级舱获救比例还是高一些

```
# 登船港口和获救的关系
fig=plt.figure()
fig.set(alpha=0.2)
s_0 = data.Embarked[data.Survived==0].value_counts()
s_1 = data.Embarked[data.Survived==1].value_counts()
df = pd.DataFrame({"获救":s_1, "未获救":s_0})
df.plot(kind="bar", stacked=True)
plt.title("各港口登陆乘客获救情况")
plt.xlabel("登陆港口")
plt.ylabel("人数")
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



登陆港口似乎和获救与否没有直接关系，但C港（法国瑟堡）获救率稍高，法国人更会逃生?

```
# 堂兄弟姐妹
g = data.groupby(by=["SibSp","Survived"])
df = pd.DataFrame(g.count()["PassengerId"])
df
```

		PassengerId
SibSp	**Survived**	
0	**0**	398
	1	210
1	**0**	97
	1	112
2	**0**	15
	1	13
3	**0**	12
	1	4
4	**0**	15
	1	3
5	**0**	5
8	**0**	7

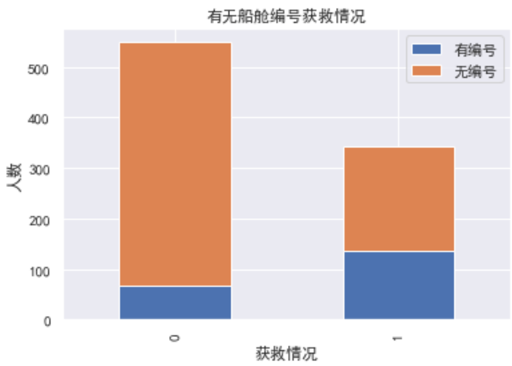
```
# 父母孩子
g = data.groupby(by=["Parch","Survived"])
df = pd.DataFrame(g.count()["PassengerId"])
df
```

		PassengerId
Parch	**Survived**	
0	**0**	445
	1	233
1	**0**	53
	1	65
2	**0**	40
	1	40
3	**0**	2
	1	3
4	**0**	4
5	**0**	4
	1	1
6	**0**	1

亲属的个数和获救没有明显的规律。似乎兄弟姐妹超过4个或父母孩子超过3个的几乎全部遇难。以后旅游不要全家倾巢出动?

```
# 船舱编号和获救的关系
fig=plt.figure()
fig.set(alpha=0.2)
s_c = data.Survived[data.Cabin=="Yes"].value_counts()
s_nc = data.Survived[data.Cabin=="No"].value_counts()
df = pd.DataFrame({"有编号":s_c, "无编号":s_nc})
df.plot(kind="bar", stacked=True)
plt.title("有无船舱编号获救情况")
plt.xlabel("获救情况")
plt.ylabel("人数")
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



似乎有编号的获救比例明显要高一些。信息健全的人应该比盲流社会地位要高一些。

6-5 数据转换

- 特征选取：哪些特征是不需要的，例如，PassengerId、Name、Ticket
- 文本字段的处理：Sex、Embarked、Cabin
- 离散化：Age、Fare
- 独热编码：one-hot

```
# 读入数据
data = pd.read_csv("data/titan.csv")
data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	No	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	Yes	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	No	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	Yes	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	No	S

```
# Sex特征
#data.Sex.value_counts()
d_sex = pd.get_dummies(data.Sex, prefix="Sex")
# Cabin特征
d_cabin = pd.get_dummies(data.Cabin, prefix="Cabin")
# Embarked特征
d_embarked = pd.get_dummies(data.Embarked, prefix="Embarked")
# Pclass特征
d_pclass = pd.get_dummies(data.Pclass, prefix="Pclass")
d_pclass.head()
```

	Pclass_1	Pclass_2	Pclass_3
0	0	0	1
1	1	0	0
2	0	0	1
3	1	0	0
4	0	0	1

```
# 将新生成的独热字段加入到数据集中
df = pd.concat([data, d_sex, d_cabin, d_embarked, d_pclass], axis=1)
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	...	Sex_female
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	...	0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	...	1
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	...	1
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	...	1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	...	0

5 rows × 22 columns

```
# 删除数据集中原有的Sex、Cabin、Embarked、Pclass特征以及不需要的PassengerId、Name、Ticket特征
df.drop(["Sex","Cabin","Embarked","Pclass","PassengerId","Name","Ticket"], axis=1, inplace=True)
df.head()
```

	Survived	Age	SibSp	Parch	Fare	Sex_female	Sex_male	Cabin_No	Cabin_Yes	Embarked_C	Embarked_(
0	0	22.0	1	0	7.2500	0	1	1	0	0	0
1	1	38.0	1	0	71.2833	1	0	0	1	1	0
2	1	26.0	0	0	7.9250	1	0	1	0	0	0
3	1	35.0	1	0	53.1000	1	0	0	1	0	0
4	0	35.0	0	0	8.0500	0	1	1	0	0	0

下面我们有两种处理方式：

- Age和Fare特征量级较大，可以先作标准化处理
- Age和Fare作离散化处理（Age可以分为老中青幼， Fare可以分为高低）

```
# Age特征离散化
age_c = pd.cut(df.Age, bins=[0,15,30,60,80],labels=["Child", "Youth", "Middle", "Old"])
# one-hot编码
d_age = pd.get_dummies(age_c, prefix="Age")
```

```python
# Fare字段离散化
fare_c = pd.cut(df.Fare, bins=[0,15,100,1000], labels=["Low","Mid","High"])
#fare_c.value_counts()
# one-hot编码
d_fare = pd.get_dummies(fare_c, prefix="Fare")
```

```python
# 对数据集重新整理
df = pd.concat([df,d_age,d_fare], axis=1)
df.drop(["Age","Fare"], axis=1, inplace=True)
```

```python
# 对于Sibsp和Parch特征处理为"有"和"无"两种情况
df.loc[df.SibSp>0,"SibSp"] = 1
df.loc[df.Parch>0, "Parch"] = 1
```

```python
df.to_csv("data/titan_clean.csv", index=False)
```

```python
df.head()
```

	Survived	SibSp	Parch	Sex_female	Sex_male	Cabin_No	Cabin_Yes	Embarked_C	Embarked_Q	Embarked_S
0	0	1	0	0	1	1	0	0	0	1
1	1	1	0	1	0	0	1	1	0	0
2	1	0	0	1	0	1	0	0	0	1
3	1	1	0	1	0	0	1	0	0	1
4	0	0	0	0	1	1	0	0	0	1

经过以上的数据变换，所有数据都变成0和1两个数值，可以用来作为机器学习模型训练的数据，来预测乘客是否会存活。

6-6 机器学习预测初步

```python
# 读入数据
import pandas as pd
df = pd.read_csv("data/titan_clean.csv")
df.head()
```

	Survived	SibSp	Parch	Sex_female	Sex_male	Cabin_No	Cabin_Yes	Embarked_C	Embarked_Q	Embarked_S
0	0	1	0	0	1	1	0	0	0	1
1	1	1	0	1	0	0	1	1	0	0
2	1	0	0	1	0	1	0	0	0	1
3	1	1	0	1	0	0	1	0	0	1
4	0	0	0	0	1	1	0	0	0	1

```python
# 分离数据和目标（X和y）
X = df.iloc[:,1:].values
y = df.iloc[:,0].values
```

```python
# 划分训练集和测试集
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=33)
```

```python
# KNN算法
from sklearn.metrics import f1_score
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("KNN:", knn.score(X_test, y_test))
print("F1-Score:", f1_score(y_test, y_pred))
```

```
KNN: 0.8295964125560538
F1-Score: 0.7738095238095237
```

```python
# 逻辑回归分类
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=1.0, penalty="l2", tol=1e-6, solver="lbfgs")
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print("LR:", lr.score(X_test, y_test))
print("F1-Score:", f1_score(y_test, y_pred))
```

```
LR: 0.8385650224215246
F1-Score: 0.7906976744186045
```

```python
# 支持向量机
from sklearn.svm import SVC
svc = SVC(gamma=0.8)
svc.fit(X_train, y_train)
y_pred = svc.predict(X_test)
print("SVM:", svc.score(X_test, y_test))
print("F1-Score:", f1_score(y_test, y_pred))
```

```
SVM: 0.8565022421524664
F1-Score: 0.8072289156626504
```

```python
# 随机森林
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=300)
rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_test)
print("RandomForest:", rfc.score(X_test, y_test))
print("F1-Score:", f1_score(y_test, y_pred))
```

```
RandomForest: 0.8430493273542601
F1-Score: 0.7928994082840237
```