

1

Kaggle案例泰坦尼克号生存预测分析

故事描述

大家都熟悉的Jack and Rose的故事，豪华游艇倒了，大家都惊恐逃生，可是救生艇的数量有限副船长说lady and kid first！，所以是否获救其实并非随机，而是基于一些背景有先后顺序的。

训练和测试数据是一些乘客的个人信息以及存活状况，要尝试根据它生成合适的模型并预测其他人的存活状况。

这是一个二分类问题，很多分类算法都可以解决。

contents



泰坦尼克号生存预测分析

1. 数据预处理读取数据
2. 数据的简单描述性分析
- 3 通过可视化的方式深入了解数据
4. 查看每一个属性与获救情况的可视化
- 5 数据缺失值处理, one-hot处理, 标准化处理
6. 使用模型进行训练---train.csv
7. 用模型进行预测---test.csv

contents



泰坦尼克号生存预测分析

通过可视化的方式深入了解数据

获救情况人数可视化

乘客等级分布可视化

按年龄看获救分布可视化

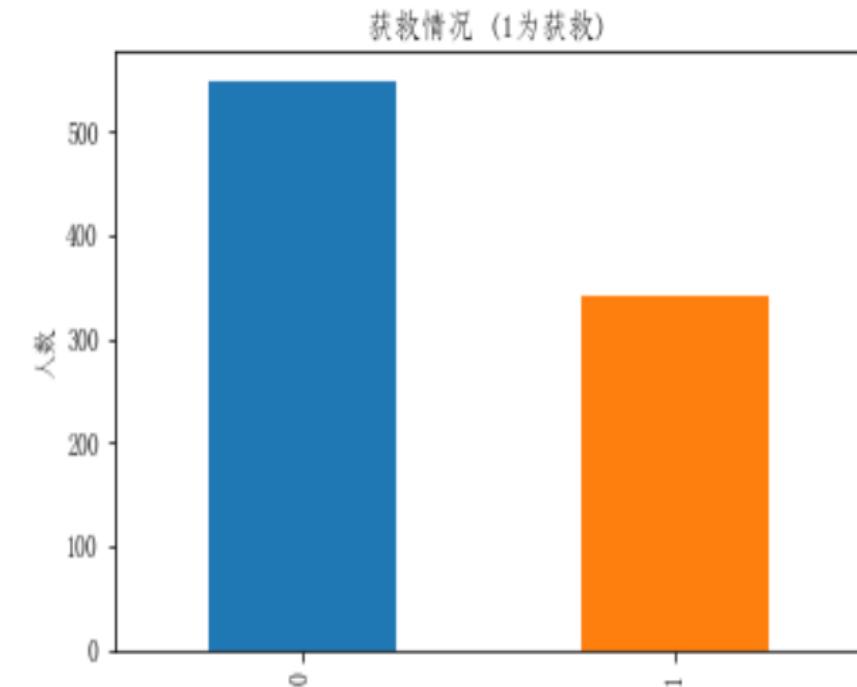
各等级的乘客年龄分布

各登船口岸上船人数可视化

获救情况人数可视化

```
import matplotlib.pyplot as plt
from pylab import mpl
mpl.rcParams['font.sans-serif'] = ['FangSong'] # 指定默认字体
mpl.rcParams['axes.unicode_minus'] = False

fig = plt.figure()
fig.set(alpha=0.2) # 设定图表颜色alpha参数
data_train.Survived.value_counts().plot(kind='bar')
plt.title("获救情况 (1为获救)")
plt.ylabel("人数")
plt.show()
```

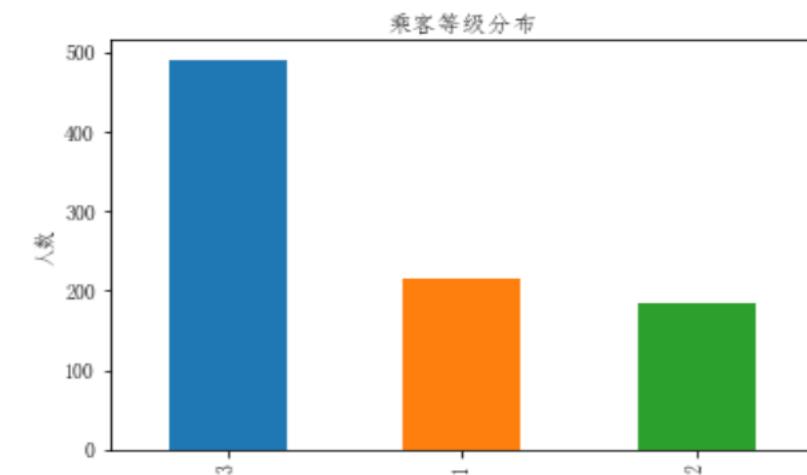


分析结果：被救的人300多点，不到半数

乘客等级分布可视化

使用DataFrame的plot方法绘制图像会按照数据的每一列绘制一条曲线，参数中的columns就是列的名称而index本来是DataFrame的行名称。图形绘制成功之后还会按照列的名称绘制图例，这个功能确实是比较赞的。如果使用matplotlib的基本绘制功能，图例的添加还需要自己额外处理。看来，数据的规整化不仅仅是为了向量化以及计算加速做准备，而且为数据的可视化提供了不少便捷的方法。

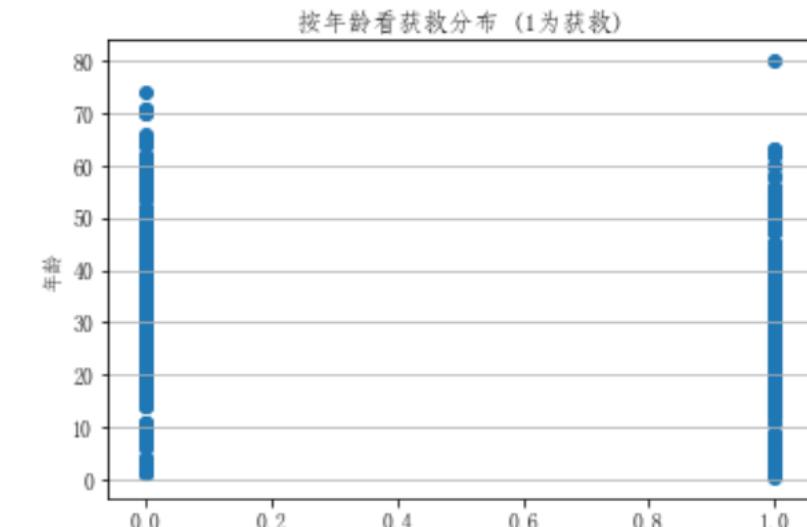
```
data_train.Pclass.value_counts().plot(kind="bar")
plt.ylabel("人数")
plt.title("乘客等级分布")
plt.show()
```



分析结果：三等舱乘客特别多

按年龄看获救分布可视化 需要传入columns方向的索引

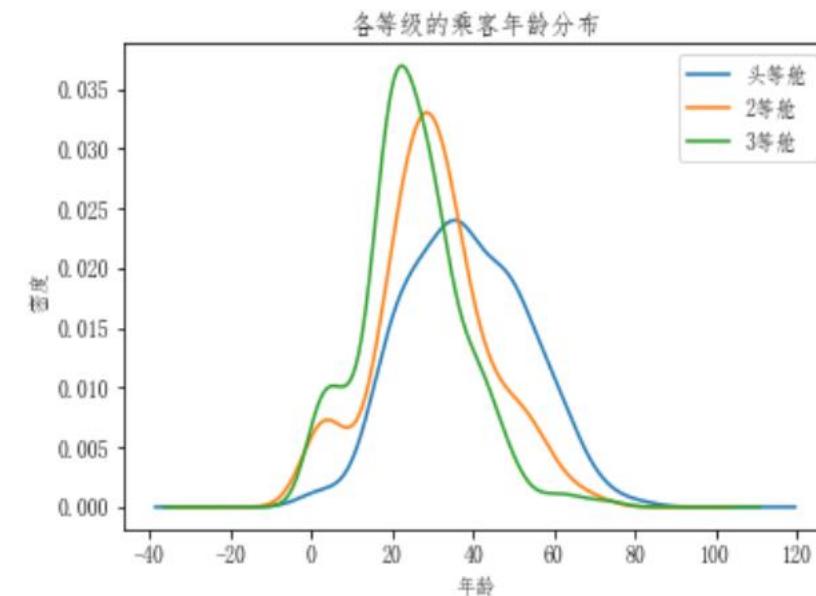
```
plt.scatter(data_train.Survived, data_train.Age)
plt.ylabel("年龄")
plt.grid(b=True, which='major', axis='y')
plt.title("按年龄看获救分布 (1为获救)")
plt.show()
```



分析结果：遇难和获救的人年龄似乎跨度都很广；

各等级的乘客年龄分布

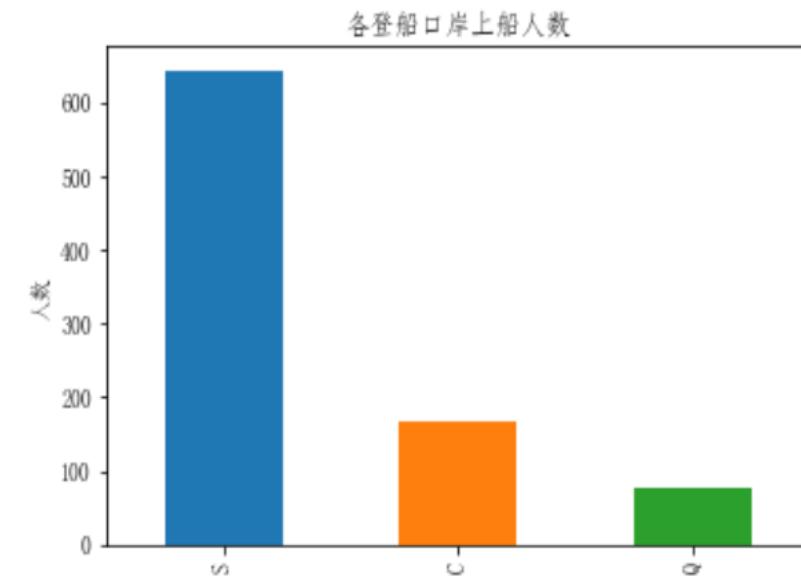
```
data_train.Age[data_train.Pclass == 1].plot(kind='kde')
data_train.Age[data_train.Pclass == 2].plot(kind='kde')
data_train.Age[data_train.Pclass == 3].plot(kind='kde')
plt.xlabel("年龄") # plots an axis label
plt.ylabel("密度")
plt.title("各等级的乘客年龄分布")
plt.legend(['头等舱', '2等舱', '3等舱'], loc='best')
plt.show()
```



分析结果：3个不同的舱年龄总体趋势一致，2/3等舱乘客20岁多点的人最多，1等舱40岁左右的最多

各登船口岸上船人数可视化

```
data_train.Embarked.value_counts().plot(kind='bar')  
plt.title("各登船口岸上船人数")  
plt.ylabel("人数")  
plt.show()
```



分析结果：登船港口人数按照S、C、Q递减，而且S远多于另外两个港口。

contents



泰坦尼克号生存预测分析

查看每一个属性与获救情况的可视化

各乘客等级的获救情况

看看各性别的获救情况

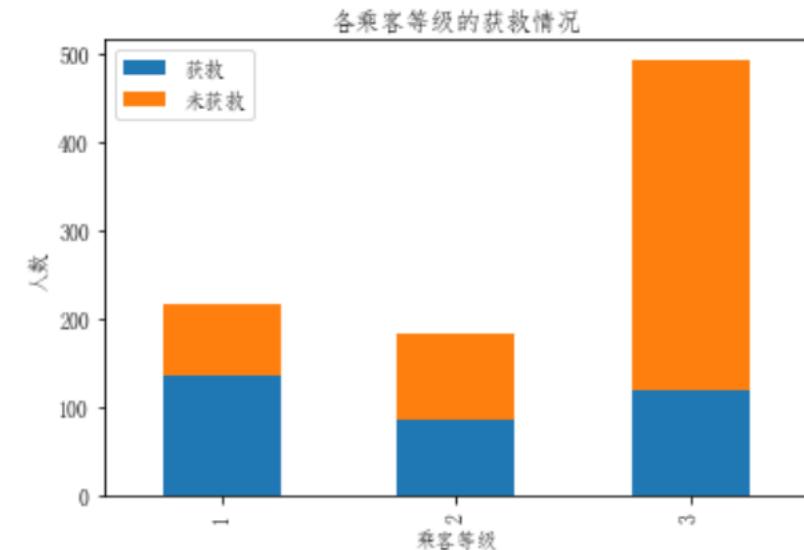
堂兄弟和父母字段对于获救情况分析

分析cabin这个值的有无，对于survival的分布状况

各乘客等级的获救情况

```
#看看各乘客等级的获救情况
fig = plt.figure()
fig.set(alpha=0.2) # 设定图表颜色alpha参数

Survived_0 = data_train.Pclass[data_train.Survived == 0].value_counts()
Survived_1 = data_train.Pclass[data_train.Survived == 1].value_counts()
df=pd.DataFrame({'获救':Survived_1, '未获救':Survived_0})
df.plot(kind='bar', stacked=True)
plt.title("各乘客等级的获救情况")
plt.xlabel("乘客等级")
plt.ylabel("人数")
plt.show()
```



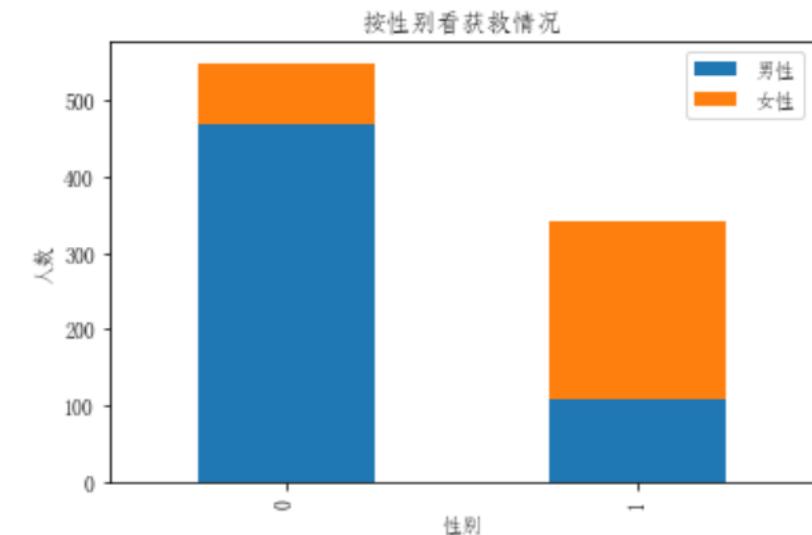
分析结果：明显等级为1的乘客，获救的概率高很多。乘客等级要作为重要特征加入最后的模型之中

看看各性别的获救情况

```
#看看各性别的获救情况
fig = plt.figure()
fig.set(alpha=0.2) # 设定图表颜色alpha参数

Survived_m = data_train.Survived[data_train.Sex == 'male'].value_counts()
Survived_f = data_train.Survived[data_train.Sex == 'female'].value_counts()
df=pd.DataFrame({u'男性':Survived_m, u'女性':Survived_f})
df.plot(kind='bar', stacked=True)
plt.title(u"按性别看获救情况")
plt.xlabel(u"性别")
plt.ylabel(u"人数")
plt.show()
```

<matplotlib.figure.Figure at 0x178450ab3c8>



分析结果：女性获救概率比较高。性别也要作为重要特征加入最后的模型之中。

堂兄弟和父母获救情况分析

```
g = data_train.groupby(['SibSp', 'Survived'])  
df = pd.DataFrame(g.count()['PassengerId'])  
df
```

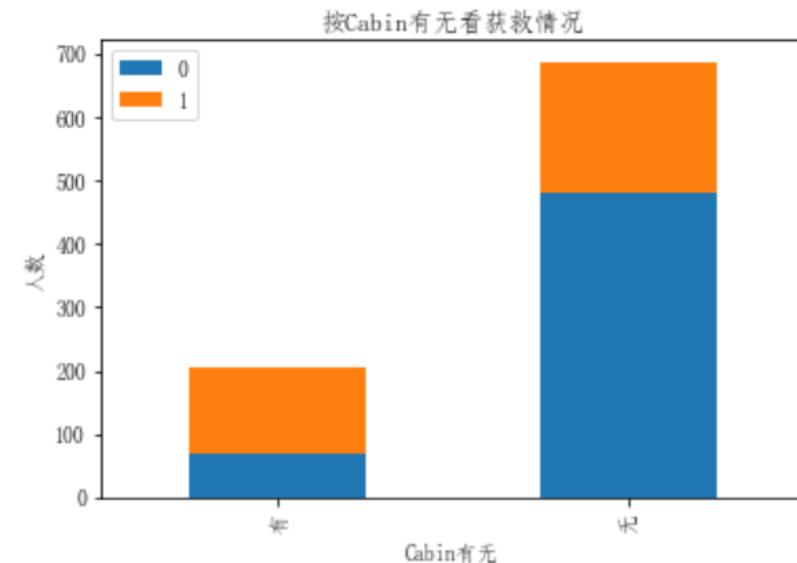
```
g = data_train.groupby(['Parch', 'Survived'])  
df = pd.DataFrame(g.count()['PassengerId'])  
df
```

分析结果：可作为备选特征

分析cabin这个值的有无，对于survival的分布状况

```
#cabin的值计数太分散了，绝大多数Cabin值只出现一次。感觉上作为类目，加入特征未必会有效
#那我们一起看看这个值的有无，对于survival的分布状况，影响如何吧
fig = plt.figure()
fig.set(alpha=0.2) # 设定图表颜色alpha参数

Survived_cabin = data_train.Survived[pd.notnull(data_train.Cabin)].value_counts()
Survived_nocabin = data_train.Survived[pd.isnull(data_train.Cabin)].value_counts()
df=pd.DataFrame({'有':Survived_cabin, '无':Survived_nocabin}).transpose()
df.plot(kind='bar', stacked=True)
plt.title("按Cabin有无看获救情况")
plt.xlabel("Cabin有无")
plt.ylabel("人数")
plt.show()
```



分析结果：有Cabin记录的似乎获救概率稍高一些

contents



泰坦尼克号生存预测分析

数据预处理

数据缺失值处理 : age, Cabin, Embarked

数据one-hot处理

数据标准化处理

```
#补充Age的缺失值
```

```
#按Cabin有无数据，将这个属性处理成Yes和No两种类型
```

```
#补充Age的缺失值
```

```
data_train['Age']=data_train['Age'].fillna(data_train['Age'].mean())
```

```
#按Cabin有无数据，将这个属性处理成Yes和No两种类型
```

```
def set_Cabin_type(df):
    df.loc[ (df.Cabin.notnull()), 'Cabin' ] = "Yes"
    df.loc[ (df.Cabin.isnull()), 'Cabin' ] = "No"
    return df
data_train = set_Cabin_type(data_train)
```

#补充Age的缺失值

#按Cabin有无数据，将这个属性处理成Yes和No两种类型

#对Embarked进行填充数据

#补充Age的缺失值

```
data_train['Age']=data_train['Age'].fillna(data_train['Age'].mean())
```

#按Cabin有无数据，将这个属性处理成Yes和No两种类型

```
def set_Cabin_type(df):
    df.loc[ (df.Cabin.notnull()), 'Cabin' ] = "Yes"
    df.loc[ (df.Cabin.isnull()), 'Cabin' ] = "No"
    return df
```

```
data_train = set_Cabin_type(data_train)
```

#对Embarked进行填充数据

```
data_train['Embarked']=data_train['Embarked'].fillna('S')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId   891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            891 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          891 non-null object
Embarked        891 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

```
# 因为逻辑回归建模时，需要输入的特征都是数值型特征
# 我们先对类目型的特征离散/因子化
# 以Cabin为例，原本一个属性维度，因为其取值可以是['yes', 'no']，而将其平展开为'Cabin_yes', 'Cabin_no'两个属性
# 原本Cabin取值为yes的，在此处的'Cabin_yes'下取值为1，在'Cabin_no'下取值为0
# 原本Cabin取值为no的，在此处的'Cabin_yes'下取值为0，在'Cabin_no'下取值为1
# 我们使用pandas的get_dummies来完成这个工作，并拼接在原来的data_train之上，如下所示
dummies_Cabin = pd.get_dummies(data_train['Cabin'], prefix= 'Cabin')

dummies_Embarked = pd.get_dummies(data_train['Embarked'], prefix= 'Embarked')

dummies_Sex = pd.get_dummies(data_train['Sex'], prefix= 'Sex')

dummies_Pclass = pd.get_dummies(data_train['Pclass'], prefix= 'Pclass')

df = pd.concat([data_train, dummies_Cabin, dummies_Embarked, dummies_Sex, dummies_Pclass], axis=1)
df.drop(['Pclass', 'Name', 'Sex', 'Ticket', 'Cabin', 'Embarked'], axis=1, inplace=True)
df.head(10)
```

我们还得做一些处理，仔细看看Age和Fare两个属性，乘客的数值幅度变化太大，进行标准差处理。

```
a=df.Age
df['Age_scaled'] = (a - a.mean()) / (a.std())
df=df.drop('Age',axis=1)

b=df.Fare
df['Fare_scaled'] = (b - b.mean()) / (b.std())
df=df.drop('Fare',axis=1)
df.head(10)
```

contents



泰坦尼克号生存预测分析

数据建模

逻辑回归



The screenshot shows the official scikit-learn website. At the top left is the logo "scikit learn". To its right is a navigation bar with links: Home, Installation, Documentation (with a dropdown arrow), and Examples. Below the navigation bar is a large blue banner featuring the text "scikit-learn" in large white letters, followed by "Machine Learning in Python" in smaller white letters. To the right of this text is a bulleted list of features:

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.

Algorithms: SVM, nearest neighbors, random forest, ...

— Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, ridge regression, Lasso, ...

— Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, ...

— Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Algorithms: PCA, feature selection, non-negative matrix factorization.

— Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning

Modules: grid search, cross validation, metrics.

— Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.

Modules: preprocessing, feature extraction.

— Examples



分类

确定对象属于哪个类别。

应用: 垃圾邮件检测, 图像识别。

算法: SVM, 最近邻居, 随机森林,

[—例子](#)

回归

预测与对象关联的连续值属性。

应用: 药物反应, 股票价格。

算法: SVR, 岭回归, 套索, [—例子](#)

聚类

将类似对象自动分组为集合。

应用程序: 客户细分, 分组实验结果

算法: k-Means, 谱聚类, 均值漂移,

[—例子](#)

维度降低

减少要考虑的随机变量的数量。

应用: 可视化, 提高效率

算法: PCA, 特征选择, 非负矩阵分解。

[—例子](#)

型号选择

比较, 验证和选择参数和模型。

目标: 通过参数调整提高准确性

模块: 网格搜索, 交叉验证, 指标。 [—例子](#)

预处理

特征提取和规范化。

应用程序: 转换输入数据 (如文本) 以用于机器学习算法。

模块: 预处理, 特征提取。

[—例子](#)

我们把需要的feature字段取出来，转成numpy格式，使用scikit-learn中的LogisticRegression建模。

```
# 我们把需要的feature字段取出来，转成numpy格式，使用scikit-learn中的LogisticRegression建模
from sklearn import linear_model

train_df = df.filter(regex='Survived|Age_.*|SibSp|Parch|Fare_.*|Cabin_.*|Embarked_.*|Sex_.*|Pclass_.*')
train_np = train_df.as_matrix()

# y即Survival结果
y = train_np[:, 0]
# y
# X即特征属性值
X = train_np[:, 1:]
# X
# fit到RandomForestRegressor之中
clf = linear_model.LogisticRegression(C=1.0, penalty='l1', tol=1e-6)
clf.fit(X, y)
clf
```

```
data_test = pd.read_csv("test.csv")
# 接着我们对test_data做和train_data中一致的特征变换
# 首先用同样的RandomForestRegressor模型填上丢失的年龄
data_test.loc[ (data_test.Fare.isnull()), 'Fare' ] = 0
#补充Age的缺失值
data_test['Age']=data_test['Age'].fillna(data_test['Age'].mean())
#按Cabin有无数据，将这个属性处理成Yes和No两种类型
def set_Cabin_type(df):
    df.loc[ (df.Cabin.notnull()), 'Cabin' ] = "Yes"
    df.loc[ (df.Cabin.isnull()), 'Cabin' ] = "No"
    return df
data_train = set_Cabin_type(data_test)

# one-hot编码
dummies_Cabin = pd.get_dummies(data_test['Cabin'], prefix= 'Cabin')
dummies_Embarked = pd.get_dummies(data_test['Embarked'], prefix= 'Embarked')
dummies_Sex = pd.get_dummies(data_test['Sex'], prefix= 'Sex')
dummies_Pclass = pd.get_dummies(data_test['Pclass'], prefix= 'Pclass')

df_test = pd.concat([data_test, dummies_Cabin, dummies_Embarked, dummies_Sex, dummies_Pclass], axis=1)
df_test.drop(['Pclass', 'Name', 'Sex', 'Ticket', 'Cabin', 'Embarked'], axis=1, inplace=True)

# 标准化处理数据
a=df_test.Age
df_test['Age_scaled'] = (a - a.mean()) / (a.std())
df_test=df_test.drop('Age',axis=1)
b=df_test.Fare
df_test['Fare_scaled'] = (b - b.mean()) / (b.std())
df_test=df_test.drop('Fare',axis=1)
df_test.head(6)
```

```
test = df_test.filter(regex='Age_.*|SibSp|Parch|Fare_.*|Cabin_.*|Embarked_.*|Sex_.*|Pclass_.*')
predictions = clf.predict(test)
result = pd.DataFrame({'PassengerId':data_test['PassengerId'].values, 'Survived':predictions.astype(np.int32)})
result.to_csv("logistic_regression_predictions.csv", index=False)
```

```
pd.read_csv("logistic_regression_predictions.csv")
```