案例六：泰坦尼克号生存者预测

1. 案例目的

• 掌握随机森林模型；

• 掌握数据预处理；

• 掌握数据可视化。

2. 案例内容

泰坦尼克号沉船事故是世界上最著名的沉船事故之一。1912年4月15日，泰坦尼克号撞上冰山后沉没。这一轰动的悲剧震惊了国际社会，并导致更好的船舶安全法规。事故中导致死亡的一个原因是许多船员和乘客没有足够的救生艇。然而在被获救群体中也有一些比较幸运的因素：一些人群在事故中被救的几率高于其他人，比如妇女、儿童和上层阶级。这个Case里，我们需要分析和判断出什么样的人更容易获救。最重要的是，要利用机器学习来预测出在这场灾难中哪些人会最终获救。

本案例将对891条泰坦尼克号乘客数据（包含多个属性：姓名、性别、年龄、船票…）进行分析，首先训练随机森林模型，然后评估模型性能，最后在测试集上预测418名乘客是否幸存。

3. 案例知识点

• Python语言编程

• NumPy和Pandas库的使用

• 随机森林模型

• 数据可视化

4. 案例时长

共4学时，具体安排如下：

• 数据预处理（1学时）

• 了解数据（1.5学时）

• 处理类别型变量（0.2学时）

• 生成自变量和因变量（0.2学时）

• 拆分数据集（0.2学时）

• 构建随机森林模型（0.5学时）

• 训练模型（0.1学时）

• 评估模型（0.3学时）

5. 案例实验环境

**•操作系统：**

1）Linux Ubuntu 16.04 64位操作系统

**•软件环境：**

1）Python 3.7

2）sklearn 0.23

**•开发环境与工具：**

1）Spyder 4

6. 案例分析

本案例主要分为以下3部分：

1）数据预处理，为建立模型做准备；

2）了解数据，以确定哪些因素对存活率的影响较大；

3）建立随机森林模型预测乘客是否幸存。

7. 案例实验过程

预测泰坦尼克号生存者，可分为以下几个步骤：

1、导入库

2、导入数据集

3、数据预处理

3.1 统计缺失数据

3.2 删除无用列

3.3 转换字段类型

3.4 处理Age字段的缺失数据

3.5 处理Embarked和Fare字段的缺失数据

4、了解数据

4.1了解Pclass字段

4.2 了解Sex字段

4.3 了解Fare字段

4.4 了解Embarked字段

4.5 了解Survived字段

4.6 了解Survived字段与Pclass字段的关系

4.7 了解Survived字段与Sex字段的关系

4.8 结合Sex字段和Fare字段了解Survived字段

4.9 了解Embarked字段和Survived字段的关系

4.10 结合Pclass字段和Fare字段了解Survived字段

4.11 结合Parch字段和Fare字段了解Survived字段

5、处理类别型变量

6、生成自变量和因变量

7、拆分数据集

8、构建随机随林模型

9、训练模型

10、评估模型性能

7.1开发准备

在桌面上新建一个目录，名字叫titanic\_survival。本案例中所有的数据集和代码都放在该目录下。

7.1.1 打开Spyder软件

在桌面上点击Spyder图标打开Spyder，并把工作目录切换至titanic\_survival目录下。

7.1.2 获取数据

本案例需要1个数据集，titanic.csv。

点击右侧菜单栏中的【资源下载】按钮，复制提供的URL前缀，如https://xxx.eec-cn.com/。

将URL前缀和数据集地址拼接到一起，如[https://xxx.eec-cn.com/dataSet/systemLib/0c3e785811b1437095486e6978bc3836.csv](https://www.eec-cn.com/u/libcourse/ff80808168f14a2d0169095b254a17d3/topDetailCombigdata_zhidu/40288042780227630178202ebd3f0722/bigDataResourceLab/bigDataTestStepa_zhidu)。下载数据集，复制到刚创建的titanic\_survival目录下。

7.1.3 新建代码文件

确保Spyder的工作目录为titanic\_survival。使用Spyder新建1个Python文件，命名为predictsurvivors.py。

至此，整个案例的目录结构图1所示。

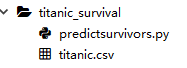


图1 案例的目录结构

7.1.5 了解数据

titanic.csv数据集包含891条数据，12个字段。这12个字段的含义分别为：

• PassengerId：乘客的编号；

• Survived：是否幸存；

• Pclass：舱位等级，1为一等舱、2为二等舱、3为三等舱；

• Name：姓名；

• Sex：性别；

• Age：年龄；

• SibSp：在船上的兄弟姐妹/配偶人数；

• Parch：在船上的父母/孩子人数；

• Ticket：船票编号；

• Fare：船票费用；

• Cabin：船舱号码；

• Embarked：登船港口， C为Cherbourg，Q为Queenstown，S为Southampton。

7.2导入库

向predictsurvivors.py添加代码。

把所有用到的库导入。

【代码7-2】predictsurvivors.py

# 导入库

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report

执行代码块。

7.3导入数据集

导入数据，并显示字段信息，显示字段值的描述。

【代码7-3】predictsurvivors.py

# 导入数据

dataset = pd.read\_csv("titanic.csv")

执行代码块。

从现在开始，建议打开Spyder的Variable explorer面板，实时查看变量的信息。

数据信息保存在dataset变量中，如图2所示。

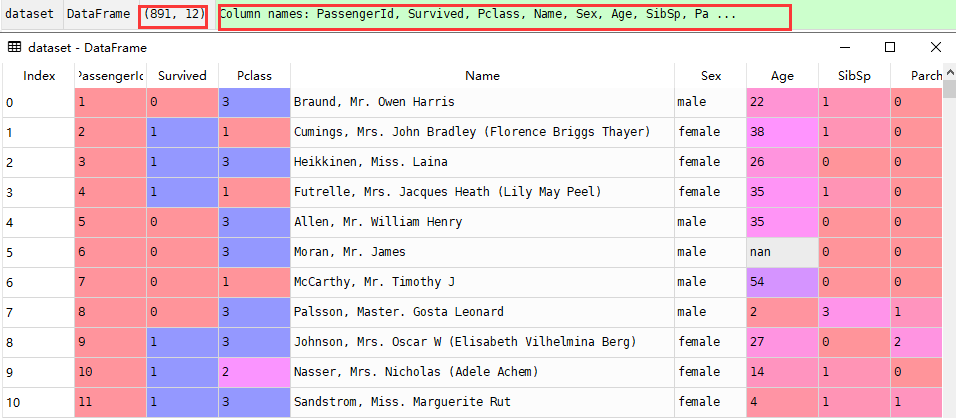


图2 数据信息保存在dataset变量中

由上图所示，可以看到数据集中有891条数据，12个字段。

7.4数据预处理

在正式开始训练模型之前，需要对数据进行预处理，为建立模型做准备。

7.4.1 统计缺失数据

统计数据集中值为空的数据。

【代码7-4-1-1】predictsurvivors.py

# 查看值为空的数据

print("数据集中的缺失数据统计\n",dataset.isnull().sum())

执行代码块。

控制台输出数据集中值为空的数据信息如图3所示。“Age”字段中有177条数据缺失，“Cabin”字段中有687条数据缺失，“Embarked”字段中有2条缺失数据。

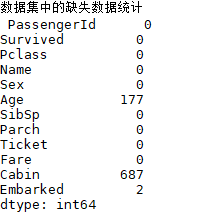


图3 训练和测试集中值为空的数据统计

计算数据集中缺失数据的百分比。

【代码7-4-1-2】predictsurvivors.py

# 计算缺失数据所占百分比

print("数据集中缺失数据占的百分比为:\n",

dataset.isnull().sum()\*100/len(dataset))

执行代码块。

打印输出数据集中缺失数据的百分比，如图4所示。

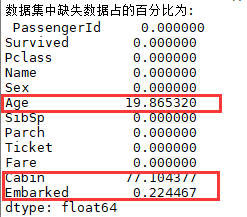


图4数据集中缺失数据的百分比

7.4.3 删除无用列

Ticket是船票编号，既不是数值型字段也不是类别型字段，应该删除；

Name是乘客姓名，既不是数值型字段也不是类别型字段，应该删除

Cabin字段缺失值比例太高，填补困难极大，建议删除。

【代码7-4-3】predictsurvivors.py

# 删除无用列

dataset.drop(["Cabin", "Ticket", "Name"],axis=1,inplace=True)

dataset.info()

执行代码块。

删除无用字段后的dataset变量中，剩余9个字段，如图5所示。

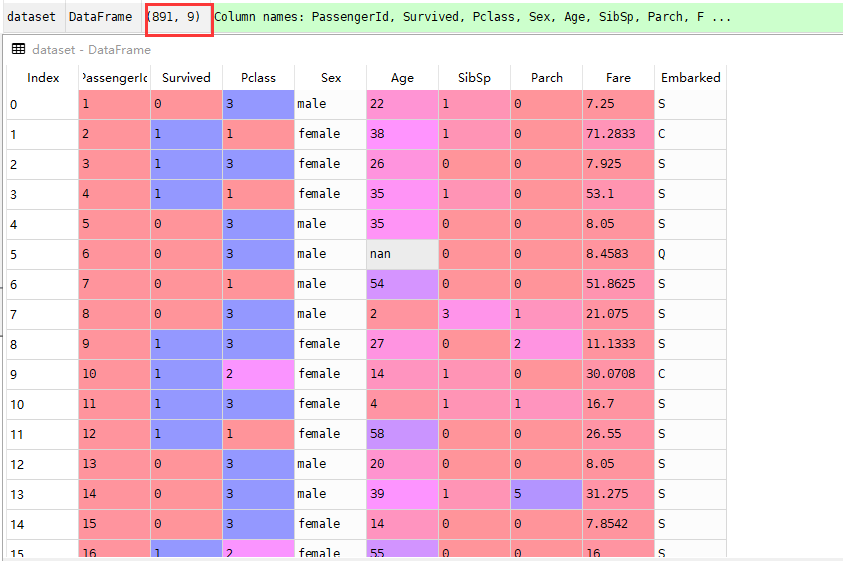


图5 删除无用字段后的dataset变量

7.4.4 转换字段类型

“Parch”、“Pclass”、“SibSp”三个字段在数据集中是数值类型。但是它本质上是类别型字段。应该将这3个字段的类型由int64转换为object。

【代码7-4-4】predictsurvivors.py

# 转换数据类型

dataset["Parch"] = dataset["Parch"].astype("object")

dataset["Pclass"] = dataset["Pclass"].astype("object")

dataset["SibSp"] = dataset["SibSp"].astype("object")

dataset.info()

执行代码块。

“Parch”、“Pclass”、“SibSp”三个字段由int64类型转换为object类型，如图6所示。

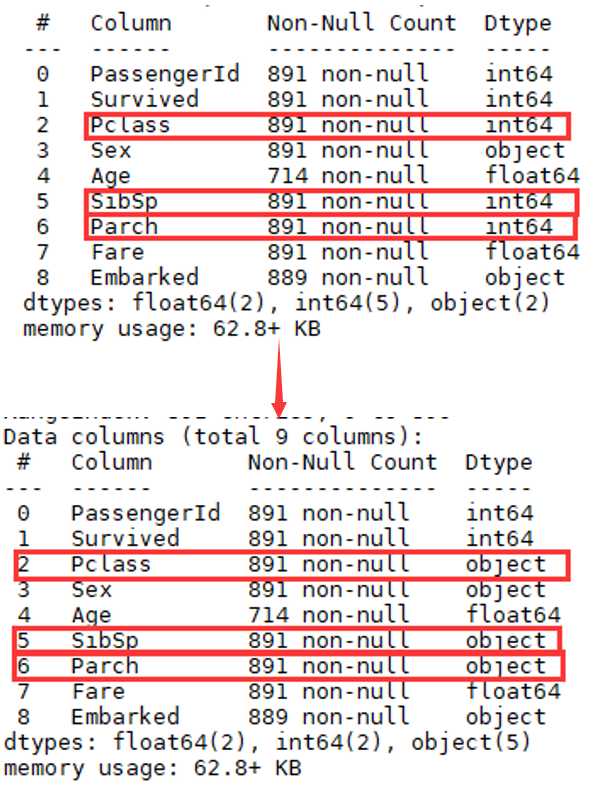


图6 数据类型转换

7.4.5 处理Age字段的缺失数据

处理Age字段的缺失数据之前，首先查看“Age”字段的分布，发现规律，使用合适的填补策略。

【代码7-4-5-1】predictsurvivors.py

# 查看Age字段的分布

print(dataset["Age"].describe())

plt.rcParams["figure.figsize"]=12,4

sns.boxplot("Age",data=dataset)

plt.title("Age distribution Boxplot")

执行代码块。

使用describe()函数查看Age字段，输出信息如图7所示。

其中，count:非空值总数；mean:非空值的平均值；std:方差；min:最小值；25%：25%分位数；50%：50%分位数（中位数）；75%:75%分位数；max:最大值；dtype:数据类型。

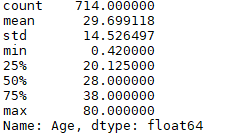


图7 Age. describe()

Boxplot可能是最常见的图形类型之一。它能够很好表示数据中的分布规律。它能显示出一组数据的最大值、最小值、中位数及上下四分位数。“Age”字段的箱型图显示如图8所示。

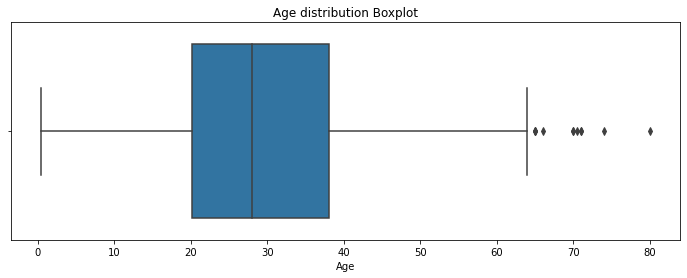


图8 Age的箱型图表示

由图7中25%分位数（20.125）和75%分位数（38）以及图8中“Age”字段的分布可知，主要年龄分布在20~38之间。

接下来，使用后describe()函数来描述数据属性，并查看年龄在20~38之间的分布。

【代码7-4-5-2】predictsurvivors.py

# 查看年龄在20~38之间的分布

print(dataset.describe())

tempdataset = dataset[(dataset['Age']>=20) & (dataset['Age']<=38)]

print(tempdataset.describe())

执行代码块。

dataset变量描述如图9所示。

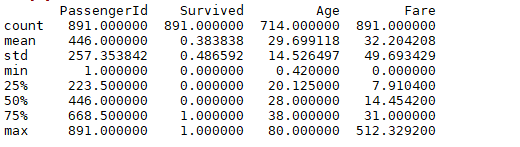


图9 dataset.describe()

“Age”在20~38之间的数据描述如图10所示。可以看到“Age”在20~38之间的均值为28.189008，50%分位数（中位数）为28。

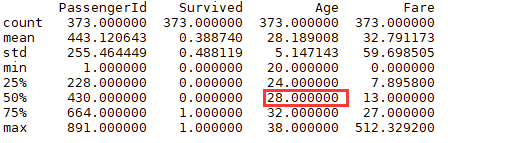


图10 “Age”在20~38之间的数据描述（tempdataset.describe()）

由上述分析结果，我们通过fillna函数填充数据集中“Age”字段缺失的值，填充为28。

【代码7-4-5-3】predictsurvivors.py

# 填充Age字段的缺失数据

print(" Age字段中缺失的数据:",dataset["Age"].fillna(28, inplace = True))

print(dataset["Age"].isnull().sum())

执行代码块。

通过isnull().sum()输出“Age”字段缺失值的数量，因为已经填充过缺失值，所以运行后，控制台输出“0”，现在“Age”字段没有缺失的值。

7.4.6 处理Embarked和Fare字段的缺失数据

“Embarked”和“Fare”字段中也缺失了一些值。正常来说，也需要向填充Age字段一样先探索字段的数据分布。因“Embarked”字段中“S”的数量最多，所以，用“S”填充“Embarked”字段缺失的数据。此外，用“Fare”字段的均值填充“Fare”字段缺失的数据。

同学们如果有时间可以更深入的探索“Embarked”字段和“Fare”字段，使用更好的填补策略。

【代码7-4-6】predictsurvivors.py

# 填充Embarked 和 Fare字段的缺失数据

print("\n Embarked字段统计:\n" ,dataset["Embarked"].value\_counts())

print("Embarked字段中缺失的数据:",dataset["Embarked"].fillna("S", inplace = True))

print("Fare字段的均值为：",dataset["Fare"].mean())

print("Fare字段中缺失的数据:",dataset["Fare"].fillna(dataset["Fare"].mean(), inplace = True))

print("dataset中缺失的数据:",dataset.isnull().sum().sum())

执行代码块。

运行代码后，控制台输出如图11所示的结果。可以看到原本“Embarked”字段中“S”的数量最多（红框内所示），因此选择了使用“S”填充“Embarked”字段中缺失的数据。“Fare”字段采用其均值进行填补。填补完成后，可以看到两个字段中的缺失数据为“None”，dataset中也没有了缺失值。

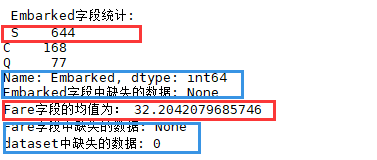


图11 “Embarked”和“Fare”字段的信息

7.5了解数据

要想创建一个强大的模型，必须先充分的了解数据的特征，才能针对数据的特性建立更好的模型，以达到更高的准确性。

7.5.1 了解Pclass字段

了解“Pclass”字段，即分别统计乘坐一等舱，乘坐二等舱，乘坐三等舱的人数。

【代码7-5-1】predictsurvivors.py

# 统计Pclass字段

plt.figure()

sns.countplot(dataset.Pclass)

plt.title("Pclass Frequency Distribution")

执行代码块。

统计结果如图12所示，可以看到乘坐三等舱的人数最多。

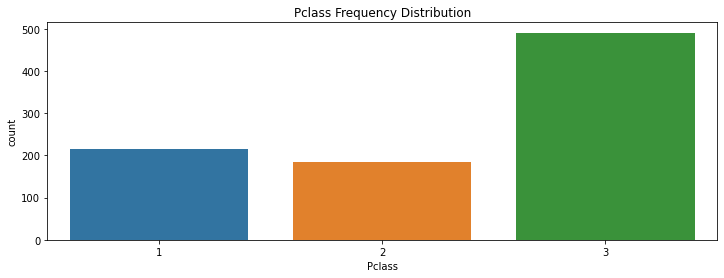


图12 Pclass字段统计

7.5.2 了解Sex字段

统计“Sex”字段，即分别统计男性乘客与女性乘客的人数。

【代码7-5-2】predictsurvivors.py

# 统计Sex字段

plt.figure()

sns.countplot(dataset.Sex)

plt.title("Sex Frequency Distribution")

执行代码块。

统计结果如图13所示，可以看到男性人数多于女性人数。

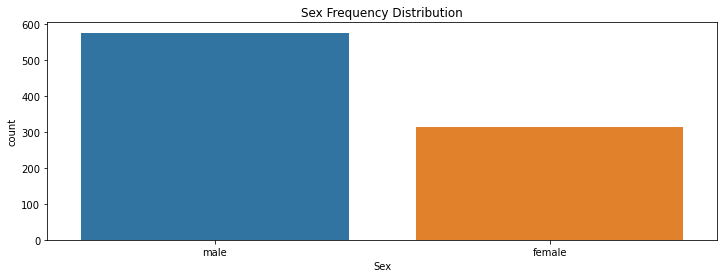


图13 Sex字段统计

7.5.3 了解Fare字段

了解“Fare”字段，即分别统计花费不同船票费用的人数。

【代码7-5-3】predictsurvivors.py

# 统计Fare字段，并画出直方图

plt.figure()

sns.distplot(dataset.Fare)

plt.title("Fare Frequency Distribution")

执行代码块。

统计结果如图14所示，可以看到花费票价较低的人数较多。

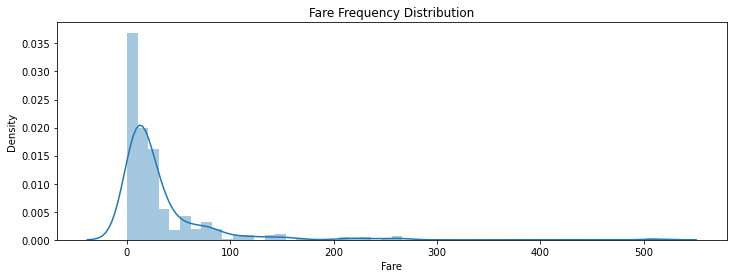


图14 Fare字段统计的直方图

7.5.4 了解Embarked字段

了解“Embarked”字段，即统计登船港口的信息。

【代码7-5-4】predictsurvivors.py

# 统计Embarked字段

plt.figure()

sns.countplot(dataset.Embarked)

plt.title("Embarked Frequency Distribution")

执行代码块。

统计结果如图15所示，可以观察到从S（Southampton）港口登船的人数最多。

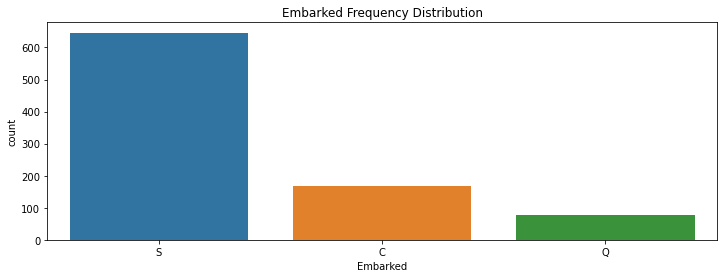


图15 Embarked字段统计

7.5.5 了解Survived字段

了解“Survived”字段，即分别统计幸存和遇难的人数。“0”表示死亡，“1”表示幸存。

【代码7-5-5】predictsurvivors.py

# 统计Survived字段

plt.figure()

sns.countplot(dataset.Survived)

plt.title("Survived Frequency Distribution")

执行代码块。

统计结果如图16所示，遇难的人数多于幸存的人数。

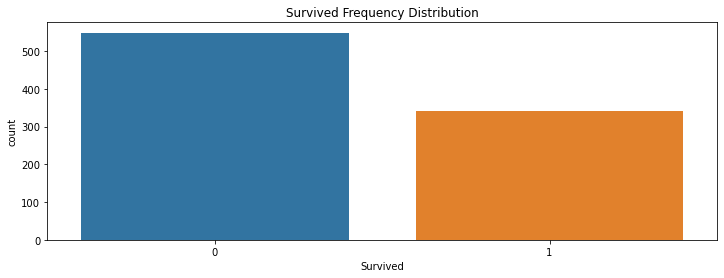


图16 Survived字段统计

7.5.6 了解Survived字段与Pclass字段的关系

了解Survived字段与Pclass字段的关系，即分别统计乘坐不同等级船舱的幸存和死亡人数。

【代码7-5-6】predictsurvivors.py

# 了解Survived字段与Pclass字段的关系

plt.figure()

sns.countplot(dataset.Pclass,hue=dataset["Survived"])

plt.title("Pclass Frequency Distribution")

执行代码块。

统计结果如图17所示，可以看到乘坐三等舱遇难的人数最多，而乘坐一等舱的幸存人数最多，遇难人数最少。

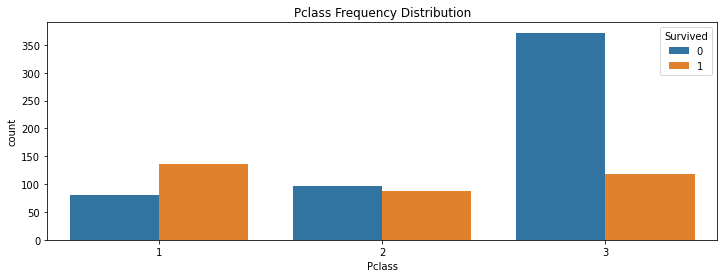


图17 Survived字段与Pclass字段的关系

7.5.7 了解Survived字段与Sex字段的关系

了解Survived字段与Sex字段的关系，即分别统计男性和女性的幸存和死亡人数。

【代码7-5-7】predictsurvivors.py

# 了解Survived字段与Sex字段的关系

plt.figure()

sns.countplot(dataset.Sex,hue=dataset["Survived"])

plt.title("Sex Frequency Distribution")

执行代码块。

统计结果如图18所示，可以看到男性的遇难人数最多，而女性的幸存人数最多。

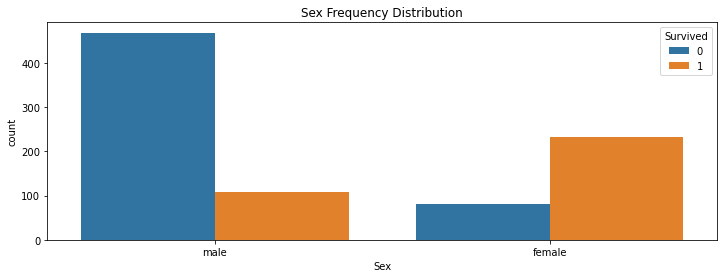


图18 Survived字段与Sex字段的关系

7.5.8 结合Sex字段和Fare字段了解Survived字段

结合Sex字段和Fare字段了解Survived字段，即结合性别和船票费用了解存活率，并以箱型图显示。

【代码7-5-8】predictsurvivors.py

# 结合Sex和Fare了解Survived

plt.figure()

sns.boxplot(x ="Survived",y="Fare",data=dataset,hue="Sex")

plt.title("Fare Frequency Distribution")

执行代码块。

统计结果如图19所示，船票费用较高的幸存人数要高于船票费用较低的人，且女性的幸存人数要高于男性的幸存人数。

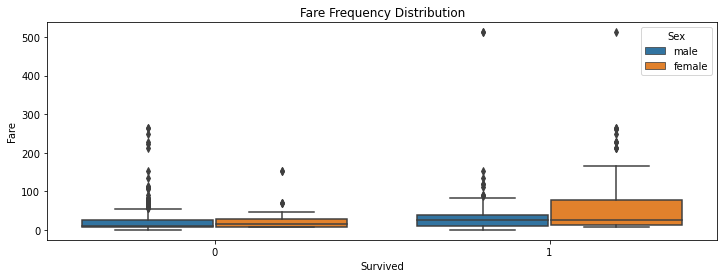


图19 结合Sex和Fare了解Survived

7.5.9 了解Embarked字段和Survived字段的关系

了解Embarked字段和Survived字段的关系，即统计从不同港口登船的人的存活率。

【代码7-5-9】predictsurvivors.py

# 了解Embarked字段和Survived字段的关系

plt.figure()

sns.countplot(dataset.Embarked,hue=dataset["Survived"])

plt.title("Embarked Frequency Distribution")

执行代码块。

统计结果如图20所示，由于从S港口登船的人数较多，所以其幸存人数和死亡人数都是最多的，C港口次之。

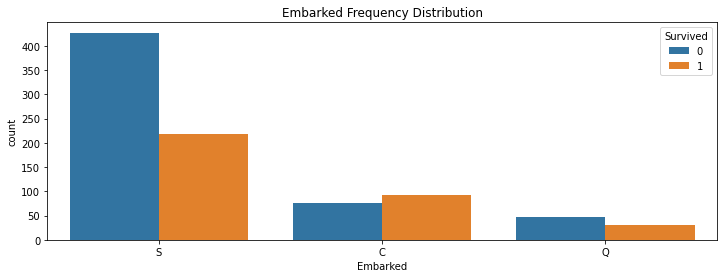


图20 Embarked字段和Survived字段的关系

7.5.10 结合Pclass字段和Fare字段了解Survived字段

结合Pclass字段和Fare字段了解Survived字段，即结合船舱等级和船票费用了解存活率，并以箱型图显示结果。

【代码7-5-10】predictsurvivors.py

# 结合Pclass字段和Fare字段了解Survived字段

plt.figure()

sns.boxplot(x ="Pclass",y="Fare",data=dataset,hue="Survived")

plt.title("Pcalss Vs Fare")

执行代码块。

统计结果如图21所示，可以观察到乘坐一等舱，且船票费用较高的幸存人数明显较多。

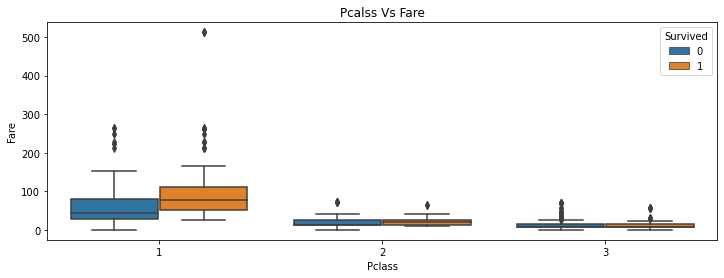


图21 结合Pclass字段和Fare字段了解Survived字段

7.5.11 结合Parch字段和Fare字段了解Survived字段

结合Parch字段和Fare字段了解Survived字段，即结合同在船上的父母或者子女人数与船票费用，观察存活率。

【代码7-5-11】predictsurvivors.py

# 结合Parch字段和Fare字段了解Survived字段

plt.figure()

sns.boxplot(x ="Parch",y="Fare",data=dataset,hue="Survived")

plt.title("Parch Vs Fare")

执行代码块。

统计结果如图22所示，可以看出有父母或者子女同行少于3人时，且花费船票费用较高的存活率较高。

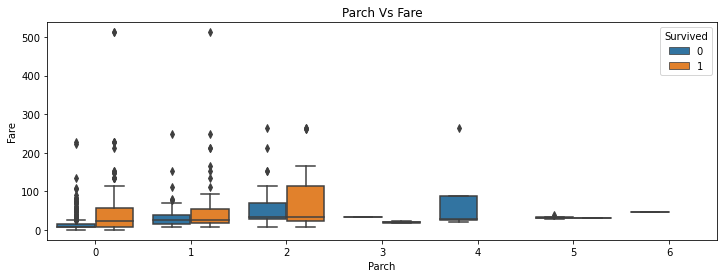


图22 结合Parch字段和Fare字段了解Survived字段

7.6处理类别型变量

类别型变量是在数据分析中十分常见的特征变量，但是在进行建模时，python不能像R那样去直接处理非数值型的变量，因此往往需要对这些类别型变量进行一系列转换。这里使用pandas中的get\_dummies方法对类别型变量做One-Hot编码（独热编码）。

【代码7-6】predictsurvivors.py

# 处理类别型变量

train = pd.get\_dummies(dataset, drop\_first = True)

执行代码块。

对类别型变量处理后的数据信息保存在train变量中，如图23所示。

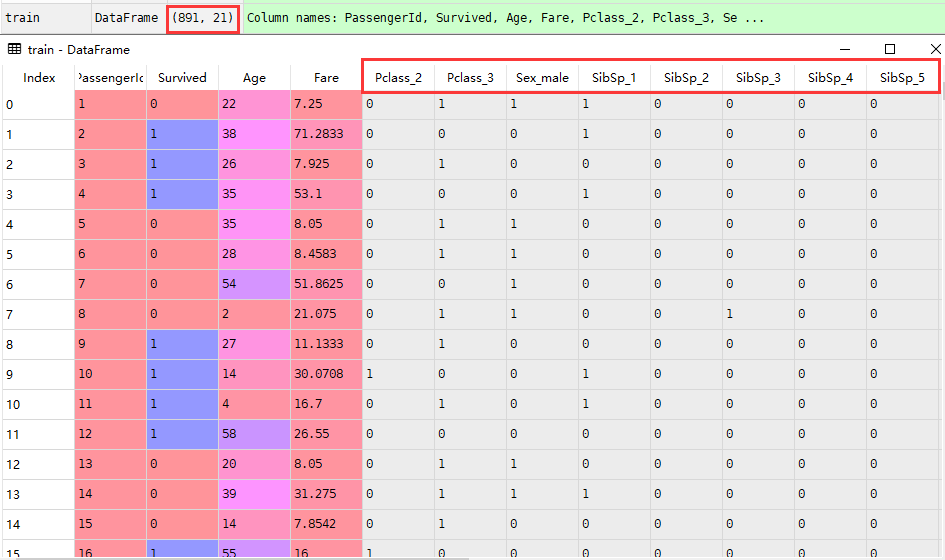


图23 对类别型变量处理后的数据信息保存在train变量中

7.7生成自变量和因变量

将“Survived”字段分离出来，作为因变量。其余字段作为自变量。

【代码7-7】predictsurvivors.py

# 生成自变量和因变量

y = train[["Survived"]].values

train.drop(['Survived'],axis=1,inplace=True)

X = train.iloc[:,:].values

执行代码块。

自变量信息保存在X变量中，因变量信息保存在y变量中，如图24所示。



图24 自变量信息保存在X变量中，因变量信息保存在y变量中

7.8拆分数据集

将数据集拆分为训练集和测试集，训练集与测试集的比例为4：1。

【代码7-8】predictsurvivors.py

# 拆分数据集

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=120)

执行代码块。

数据集划分后的数据信息分别保存在X\_train，X\_test，y\_train，y\_test四个变量中，如图25所示。

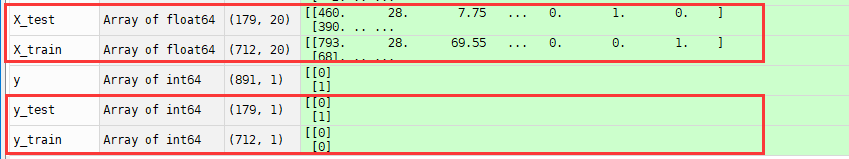


图25 数据集划分后的数据信息分别保存在X\_train,X\_test,y\_train,y\_test四个变量中

7.9构建随机森林模型

该部分的作用是构建随机森林模型，各参数含义如下：

criterion：分裂节点所用的标准，可选“gini”,“entropy”，默认“gini”；

max\_depth：树的最大深度；

max\_features：划分叶子节点，选择的最大特征数目；

max\_leaf\_nodes：最大叶子节点数；

n\_estimators：随机森林中树的个数，即学习器的个数；

random\_state：随机状态，默认由np.numpy生成。

同学们可以根据模型性能评估结果调节这些参数，以获得更好的模型。

【代码7-9】predictsurvivors.py

# 构建随机森林模型

rfc = RandomForestClassifier(criterion = 'gini',

max\_depth = 7,

max\_features = 'auto',

max\_leaf\_nodes = 20,

n\_estimators = 50,

random\_state=0)

执行代码块。

7.10训练模型

在训练集上训练随机森林模型。

【代码7-10】predictsurvivors.py

# 训练模型

rfc.fit(X\_train, y\_train)

执行代码块。

7.11评估模型性能

模型训练完成后，评估模型性能。

【代码7-11】predictsurvivors.py

# 评估模型性能

print(classification\_report(y\_test, rfc.predict(X\_test)))

执行代码块。

性能评估结果如图26所示，accuracy达到了83%，macro avg的值都在76%以上，weighted avg的值都在82%以上，模型性能还是比较不错的。

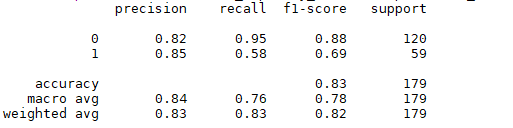


图26 性能评估结果

8. 案例代码

【案例代码】predictsurvivors.py

'''

泰坦尼克号生存者预测

'''

# 导入库

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report

# 导入数据

dataset = pd.read\_csv("titanic.csv")

# 查看值为空的数据

print("数据集中的缺失数据统计\n",dataset.isnull().sum())

# 计算缺失数据所占百分比

print("数据集中缺失数据占的百分比为:\n",

dataset.isnull().sum()\*100/len(dataset))

# 删除无用列

dataset.drop(["Cabin", "Ticket", "Name"],axis=1,inplace=True)

dataset.info()

# 转换数据类型

dataset["Parch"] = dataset["Parch"].astype("object")

dataset["Pclass"] = dataset["Pclass"].astype("object")

dataset["SibSp"] = dataset["SibSp"].astype("object")

dataset.info()

# 查看Age字段的分布

print(dataset["Age"].describe())

plt.rcParams["figure.figsize"]=12,4

sns.boxplot("Age",data=dataset)

plt.title("Age distribution Boxplot")

# 查看年龄在20~38之间的分布

print(dataset.describe())

tempdataset = dataset[(dataset['Age']>=20) & (dataset['Age']<=38)]

print(tempdataset.describe())

# 填充Age字段的缺失数据

print(" Age字段中缺失的数据:",dataset["Age"].fillna(28, inplace = True))

print(dataset["Age"].isnull().sum())

# 填充Embarked 和 Fare字段的缺失数据

print("\n Embarked字段统计:\n" ,dataset["Embarked"].value\_counts())

print("Embarked字段中缺失的数据:",dataset["Embarked"].fillna("S", inplace = True))

print("Fare字段的均值为：",dataset["Fare"].mean())

print("Fare字段中缺失的数据:",dataset["Fare"].fillna(dataset["Fare"].mean(), inplace = True))

print("dataset中缺失的数据:",dataset.isnull().sum().sum())

# 了解数据

# 统计Pclass字段

plt.figure()

sns.countplot(dataset.Pclass)

plt.title("Pclass Frequency Distribution")

# 统计Sex字段

plt.figure()

sns.countplot(dataset.Sex)

plt.title("Sex Frequency Distribution")

# 统计Fare字段，并画出直方图

plt.figure()

sns.distplot(dataset.Fare)

plt.title("Fare Frequency Distribution")

# 统计Embarked字段

plt.figure()

sns.countplot(dataset.Embarked)

plt.title("Embarked Frequency Distribution")

# 统计Survived字段

plt.figure()

sns.countplot(dataset.Survived)

plt.title("Survived Frequency Distribution")

# 了解Survived字段与Pclass字段的关系

plt.figure()

sns.countplot(dataset.Pclass,hue=dataset["Survived"])

plt.title("Pclass Frequency Distribution")

# 了解Survived字段与Sex字段的关系

plt.figure()

sns.countplot(dataset.Sex,hue=dataset["Survived"])

plt.title("Sex Frequency Distribution")

# 结合Sex和Fare了解Survived

plt.figure()

sns.boxplot(x ="Survived",y="Fare",data=dataset,hue="Sex")

plt.title("Fare Frequency Distribution")

# 了解Embarked字段和Survived字段的关系

plt.figure()

sns.countplot(dataset.Embarked,hue=dataset["Survived"])

plt.title("Embarked Frequency Distribution")

# 结合Pclass字段和Fare字段了解Survived字段

plt.figure()

sns.boxplot(x ="Pclass",y="Fare",data=dataset,hue="Survived")

plt.title("Pcalss Vs Fare")

# 结合Parch字段和Fare字段了解Survived字段

plt.figure()

sns.boxplot(x ="Parch",y="Fare",data=dataset,hue="Survived")

plt.title("Parch Vs Fare")

# 处理类别型变量

train = pd.get\_dummies(dataset, drop\_first = True)

# 生成自变量和因变量

y = train[["Survived"]].values

train.drop(['Survived'],axis=1,inplace=True)

X = train.iloc[:,:].values

# 拆分数据集

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=120)

# 构建随机森林模型

rfc = RandomForestClassifier(criterion = 'gini',

max\_depth = 7,

max\_features = 'auto',

max\_leaf\_nodes = 20,

n\_estimators = 50,

random\_state=0)

# 训练模型

rfc.fit(X\_train, y\_train)

# 评估模型性能

print(classification\_report(y\_test, rfc.predict(X\_test)))