几种不同的机器学习方法预测泰坦尼克号幸存者

Hollow Man

目录

摘要

本文基于 kaggle 上的一个著名机器学习项目 Titanic: Machine Learning from Disaster，通过数据清洗，并使用不同的机器学习方法创建模型，使得模型能够预测哪些乘客在泰坦尼克号沉船事故中幸存下来。

关键词

机器学习、分类预测、kaggle

一、项目简介

Kaggle 是一个数据分析建模的应用竞赛平台，Titanic: Machine Learning from Disaster 是其上的一个入门级机器学习数据预测的比赛项目。

泰坦尼克号的沉没是历史上最著名的海难之一。

1912 年 4 月 15 日，在她的处女航中，被认为永远不可能沉没的泰坦尼克号与冰山相撞后沉没。与此同时，不幸的是，船上没有足够的救生艇供所有人使用，导致 2224 名乘客和机组人员中的 1502 人死亡。

尽管幸存是需要一些运气的，但似乎存在一些潜在的规律，使得有些人比其他人更有可能生存。

在这一比赛中，题目要求使用乘客数据（例如姓名，年龄，性别，社会经济阶层等）来建立一个回答以下问题的预测模型：“什么样的人在泰坦尼克号沉船事故中更有可能幸存下来？”。

该比赛项目的网址请见 https://www.kaggle.com/c/titanic。

图 1 展示了 Kaggle 平台上该比赛项目的界面。

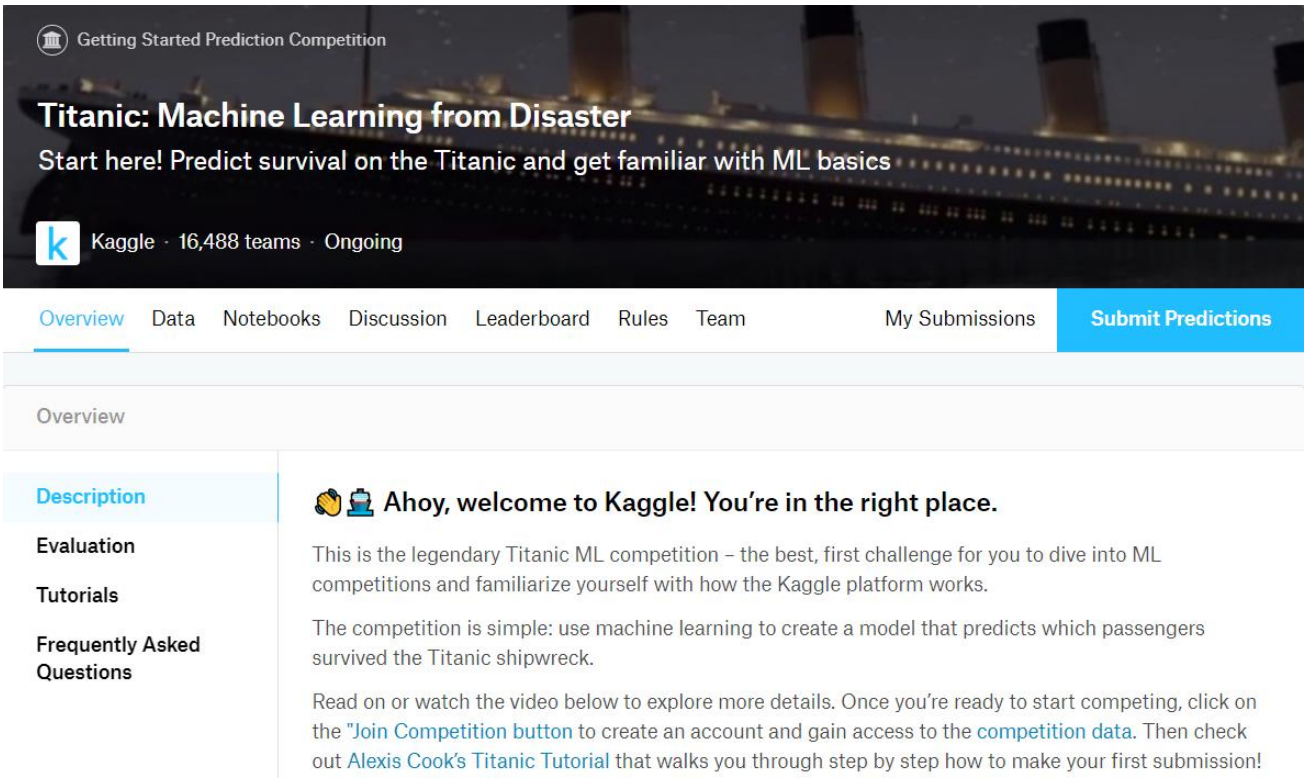

图 1　比赛项目界面

二、工作流程

1. 定义问题：在解决问题之前，我们必须要明白问题是什么。

2. 收集数据，数据清洗：是将原始数据转换为计算机可读取识别处理数据的必需过程。数据包括实现用于存储和处理的数据架构，开发用于质量和控制的数据治理标准，数据提取（即 ETL 和网络抓取）以及用于识别异常，丢失或异常数据点的数据清理。

3. 探索性分析：部署描述性和图形化统计信息以查找数据集中的潜在问题，模式，分类，相关性和比较非常重要。此外，数据分类（即定性与定量）对于理解和选择正确的假设检验或数据模型也很重要。

4. 模型数据：与描述性和推论性统计数据一样，数据建模可以汇总数据或预测未来结果。

5. 验证和实施数据模型：在根据数据子集训练模型后，是时候测试模型了。这有助于确保不会过度拟合模型或使其特定于所选子集，因为它不能准确地适合同一数据集中的另一个子集。在这一步中，我们确定我们的模型是否适合，概括或不适合我们的数据集。

6. 优化和策略：改进模型，让它更好，更强，比以前更快。

下图展示了一般数据数据分析问题的工作流程。

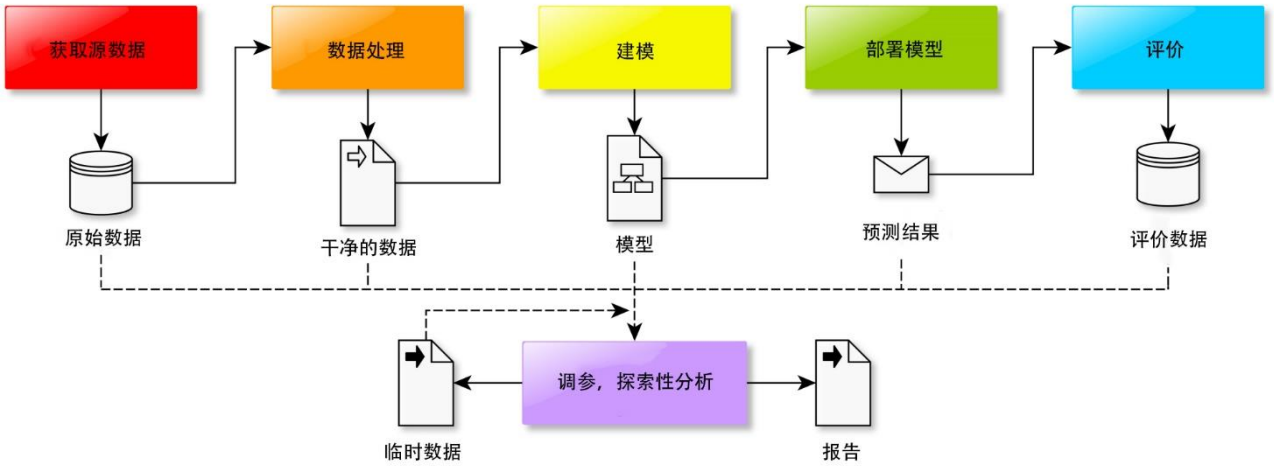

图 2　一般数据分析问题工作流程

三、使用语言和库的介绍

我们使用 Python 3 语言来进行本次项目的处理，并且使用数据科学处理库 numpy 与 scipy，具有数据处理和分析功能的 pandas 库，可视化库 matplotlib 与 seaborn，机器学习算法库 Scikit-learn。

作为专门面向机器学习的 Python 开源框架，Scikit-learn 可以在一定范围内为开发者提供非常好的帮助。它内部实现了各种各样成熟的算法，容易安装和使用，样例丰富，而且教程和文档也非常详细。

Scikit-learn 的基本功能主要被分为六大部分：分类，回归，聚类，数据降维，模型选择和数据预处理。

下面是对 Scikit-learn 的基本功能的简介：

（1）**Preprocessing 预处理**

- 应用：转换输入数据，规范化、编码化
- 模块：preprocessing，feature_extraction，transformer（转换器）

（2）**Dimensionality reduction 降维**

- 应用：Visualization（可视化），Increased efficiency（提高效率）
- 算法：主成分分析(PCA)、非负矩阵分解（NMF），feature_selection(特征选择)等

（3）**Classification 分类**

- 应用：二元分类问题、多分类问题、Image recognition 图像识别等
- 算法：逻辑回归、SVM，最近邻，随机森林，Naïve Bayes，神经网络等

（4）**Regression 回归**

- 应用：Drug response 药物反应，Stock prices 股票价格
- 算法：线性回归、SVR，ridge regression，Lasso，最小角回归（LARS）等

（5）**Clustering 聚类**

- 应用：客户细分，分组实验结果
- 算法：k-Means，spectral clustering(谱聚类)，mean-shift（均值漂移）

（6）**Model selection 模型选择**

- 目标：通过参数调整提高精度
- 模块：pipeline(流水线)，grid_search（网格搜索），cross_validation(交叉验证)，metrics（度量），learning_curve（学习曲线）

（7）、**模型融合**

- 模块：ensemble(集成学习)、

（8）、**辅助工具**

- 模块：exceptions(异常和警告)、dataset（自带数据集）、utils、sklearn.base

给定一个名为 model 的 scikit 学习器对象，可以使用以下方法：

适用于所有学习器：

model.fit ()：拟合训练数据。对于监督学习，它接受两个参数：数据 X 和标签 y（例如 model.fit（X，y））。对于无监督学习，它只接受一个参数，即数据 X（例如 model.fit（X））。

在监督学习中可用的方法：

model.predict()：给定一个经过训练的模型，预测一组新数据的标签。此方法接受一个参数，即新数据 X_new（例如 model.predict（X_new）），并返回数组中每个对象的预测标签值。

model.predict_proba ()：对于分类问题，一些估计器也提供了这种方法，它返回一个新观测值具有每个分类标签的概率。在这种情况下，

model.predict ()：返回概率最高的标签。

model.score ()：对于分类或回归问题，大多数（接近于全部）的学习器采用评分法。分数在 0 到 1 之间，分数越大表示越适合。

在无监督估计中可用的方法：

model.predict（）：聚类算法中的预测标签。

model.transform（）：给定一个无监督的模型，将新数据转换为新基础。它还接受一个参数 X_new，并返回基于无监督模型的数据的新表示。

model.fit_transform()：一些估计器实现了这种方法，它可以更有效地对相同的输入数据执行拟合和转换。

四、数据集的处理

要进行模型的建立，我们首先需要知道我们现在有哪些数据，并对其进行处理，从而方便计算机的读取与操作，为训练做准备。

该比赛提供的数据集分为两组：**训练集（train.csv）和测试集（test.csv）**，训练集通常是用来构建机器学习模型。对于训练集，该比赛项目已经提供了每个乘客是否幸存的结果。

幸存预测是基于乘客的性别和阶级等“特征”来进行的。

测试集是用来测试模型在未知数据上的表现。对于测试集，该比赛项目不提供每位乘客的基本情况。对于测试集中的每位乘客，我们需要使用训练的模型来预测他们是否在泰坦尼克号沉没中幸存了下来。

该项目还提供了 gender_submission.csv，这是通过假设所有女性乘客都可以幸存得出的结果，此文件是作为提交文件的示例。

4.1、数据集各键含义

变量	定义	值代表的含义
survival	是否幸存	0 = 否, 1 = 是
pclass	舱位等级	1 = 头等舱, 2 = 中等, 3 = 下等
sex	性别	
Age	年龄	
sibsp	其兄弟姐妹/配偶随同登船的人数	
parch	其父母/孩子随同登船的人数	
ticket	票号	
fare	票价	
cabin	船舱号	

embarked	登船港口	C = 瑟堡, Q = 皇后镇, S = 南安普敦

表 1　数据集中各变量的含义

Age：如果年龄小于 1，则年龄是小数。如果年龄是被估算的，则会以 xx.5 形式出现

sibsp：数据集定义这样的家庭关系：兄弟姐妹=兄弟，姐妹，同父异母的兄弟姐妹，配偶=丈夫，妻子（情夫/妇和未婚者被忽略）

parch：将数据集定义家庭关系以这种方式：孩子=女儿，儿子，继子女，有些孩子随同保姆旅行，因此他们的 parch 等于 0。

4.2、加载并查看数据

使用 pandas.read_csv 方法读取数据集内数据，训练集数据赋值给变量 datatrain，测试集数据赋值给变量 datatest。

下面是部分训练集内的读取数据：

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736	30.0708	NaN	C

图 3　一部分训练集内的数据展示

打印出 shape 属性，可知测试集中有 418 个数据，训练集中有 891 个数据。

打印出数据集的 isnull().sum()，发现训练集中 177 人缺少 Age 年龄信息，687 人缺少 Cabin 船舱号信息，2 人缺少 Embarked 登船港口信息，测试集中 86 人缺少 Age 年龄信息， 327 人缺少 Cabin 船舱号信息，1 人缺少 Fare 票价信息。

4.3、数据清洗

在这一步我们需要（1）纠正异常值，（2）补全缺失信息，（3）创建新的分析值，（4）将字段转换为正确的格式进行计算和显示。

更正：查看数据时，似乎没有任何异常或不可接受的数据输入。此外，我们发现我们可能在年龄和票价方面

存在潜在的异常值。但是，由于它们是合理的值，我们将等到我们完成探索性分析后确定是否应该包含或排除数据集。应该注意的是，如果它们是不合理的值，例如 age= 800 而不是 80，那么我们应该修正它。但是，当我们从原始值修改数据时，我们要谨慎使用，因为我们需要创建一个准确的模型。

补全：年龄，船舱号和登船港口中存在空值或缺失数据。丢失的值可能很糟糕，因为某些算法不知道如何处理空值因而报错。而其他模型，如决策树，可以处理空值。因此，在开始建模之前修复是很重要的，因为我们将比较和对比几个模型。有两种常用方法，要么删除记录，要么使用合理的输入填充缺失值。建议不要删除记录，尤其是大部分记录，除非它真正代表不完整的记录。相反，最好填充缺失值。定性数据的基本方法是使用众数。定量数据的基本方法是使用均值，中位数或均值+随机标准差来估算。中间方法是使用基于特定标准的基本方法；比如按类划分的平均年龄或按票价和登船港口。对于此数据集，缺失的年龄 Age 将使用中位数估算，缺失的登船港口 Embarked 将用众数填充，缺失的票价 Fare 将用中位数填充，同时我们删除了对预测幸存者无作用的 ticket 票号，大部分数据都缺失的 cabin 船舱号。

创建：特征工程是指我们使用现有功能创建新特征，来更好地体现数据含义，从而预测结果。对于本数据集，我们创建了 FamilySize 家庭大小这一特征，通过计算 SibSp 其兄弟姐妹/配偶随同登船的人数和 Parch 其父母/孩子随同登船的人数和得到值；还创建了 IsAlone 是否独自旅行这一特征，通过判断 FamilySize 家庭大小，为 0 则将该键值赋为 1，否则赋为 0；同时我们通过分离 Name 中的称谓创建了 Title 称谓这一特征，同时将出现次数较少的称谓用 Misc 代替，同时通过创建 FareBin 和 AgeBin 特征，通过 pandas.cut 与 qcut 将 Age 年龄和 fare 票价分别转换为年龄范围组 FareBin 和票价范围组 AgeBin。

转换：最后，将数据转换为计算机易于处理的格式，从而方便进行训练。

4.4、可视化分析数据

使用 seaborn 统计训练集中幸存者的人数，如图 4 所示，1 代表幸存，0 代表遇难，可以发现遇难者人数多于幸存者人数。

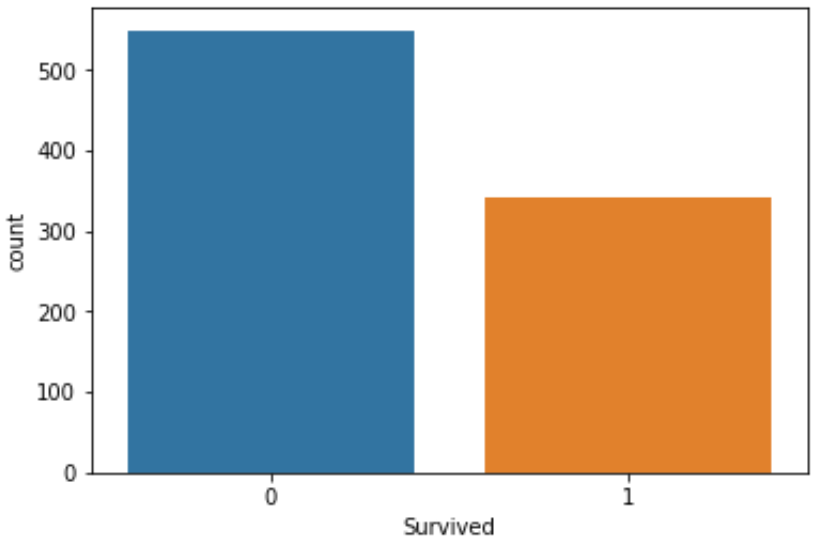

图 4　训练集中幸存和未幸存者的人数

再分析训练集中幸存与出发港口、是否独自旅行、舱位等级、性别关系，如图 5 所示，可见遇难者和幸存者相比，有向男性，独自旅行，下等舱位的偏向趋势。

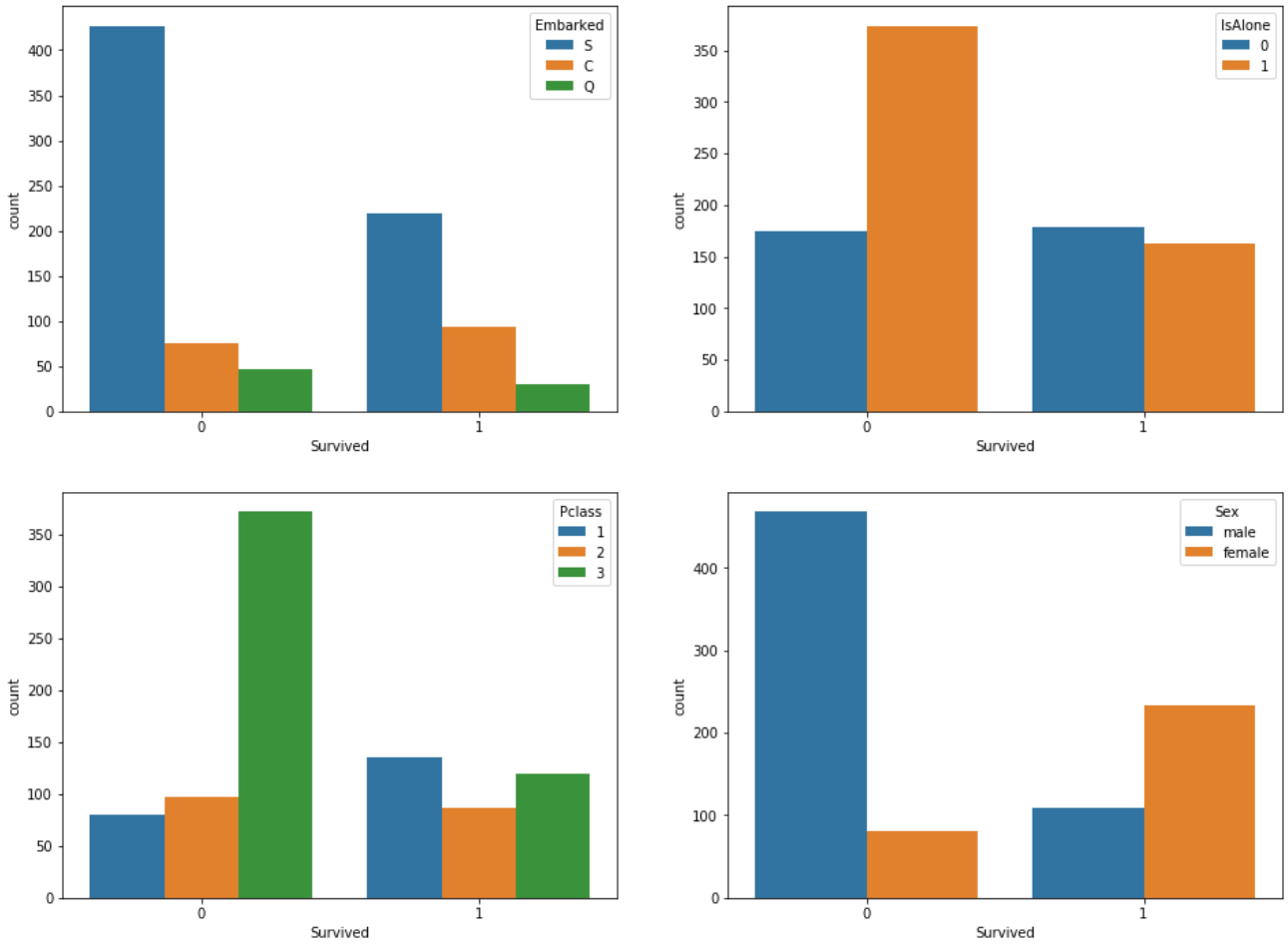

图 5　训练集中幸存与出发港口，是否独自旅行，舱位等级，性别关系统计

接下来分析舱位和票价、年龄、家庭大小关系的生存机率比较，如图 6 所示，可见是头等舱，票价越高生存几率越大，三种舱位都呈现低龄生存几率更高，且一同旅行的家庭成员数越多，生存几率越大。

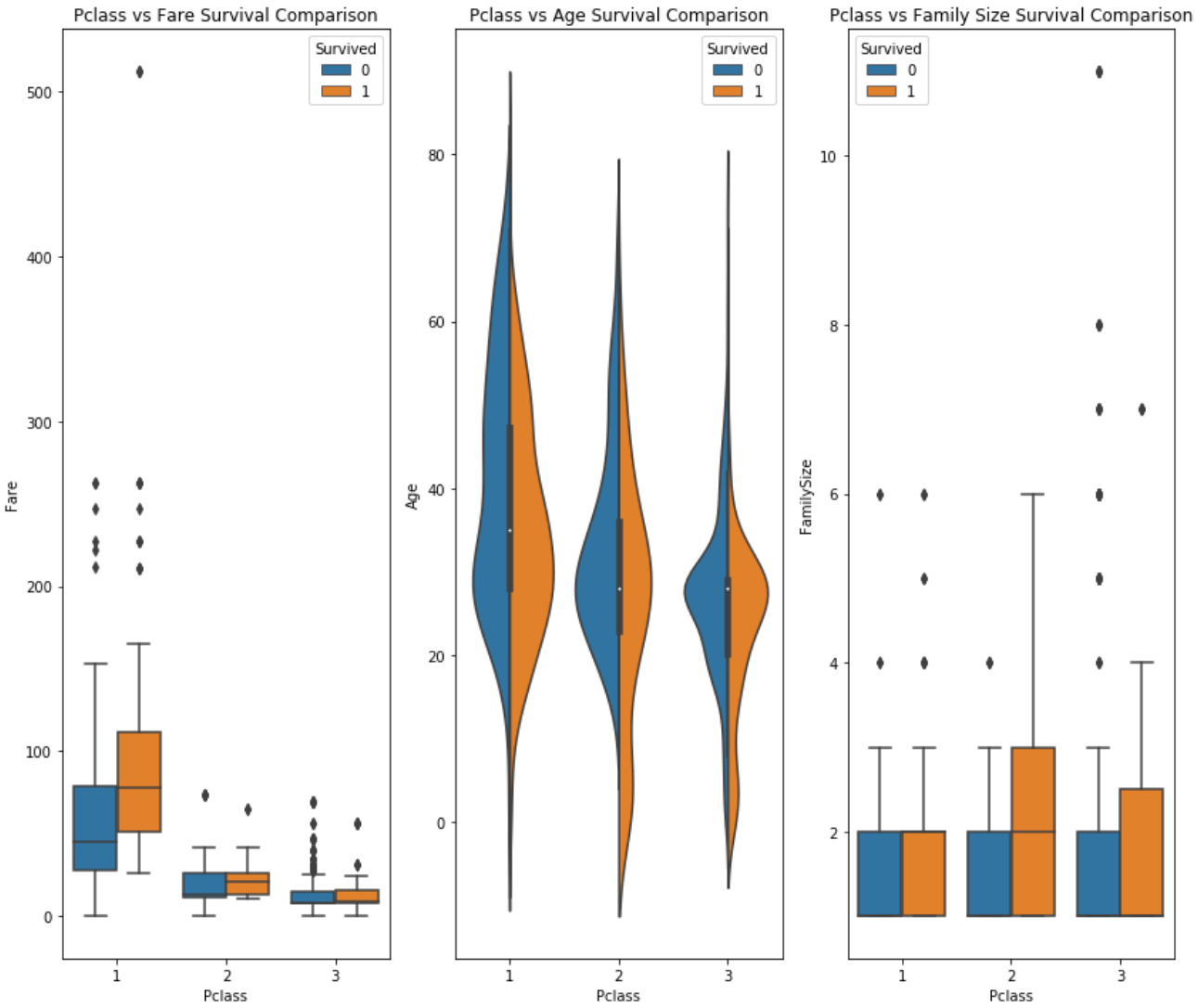

图 6　训练集中舱位和票价、年龄、家庭大小关系的生存机率统计

接下来分析幸存者和遇难者年龄分布，如图 7 所示，低龄生存几率更高，中龄人生存几率相对较低，老年人生存与遇难无明显趋向。

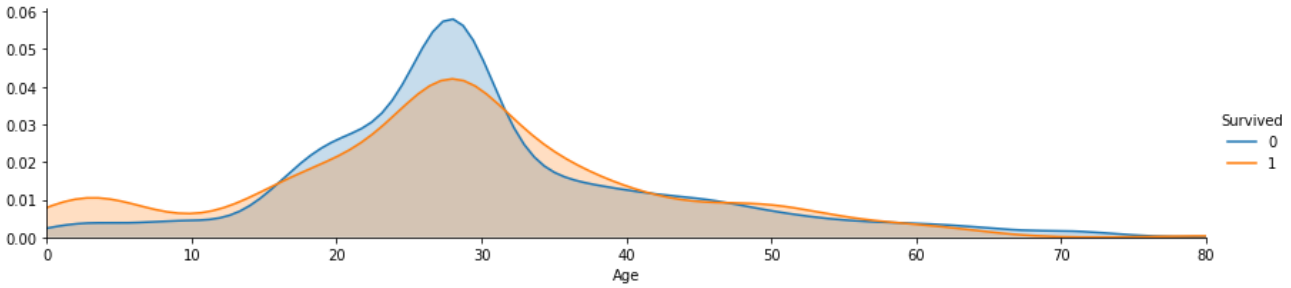

图 7　训练集中幸存者和遇难者年龄分布

最后画出各个键之间的热力图，见图 8，直观感受各变量之间的关系。

图 8　训练集中各个键之间的热力图

4.5、转换数据

最后，将数据转换为计算机易于处理的格式，从而方便进行训练。

将性别 Sex 转换为 0 和 1 的格式，0 为男性，1 为女性，再将出发港口 Embarked 的 S C Q 分别转换为 0 1 2，称谓 Title 中 Mr 标记位 1，Miss 标记位 2，Mrs 标记为 3，Master 标记为 4，Misc 标记为 5。

以上转换均用 sklearn.preprocessing 中的 LabelEncoder().fit_transform 实现，然后将 survived 作为目标值 y，其它的键值作为变量 x。同时在给定的训练集中划分出测试集和训练集，分别命名为 test1_x_dummy 和 train1_x_dummy。

五、使用机器学习算法进行训练

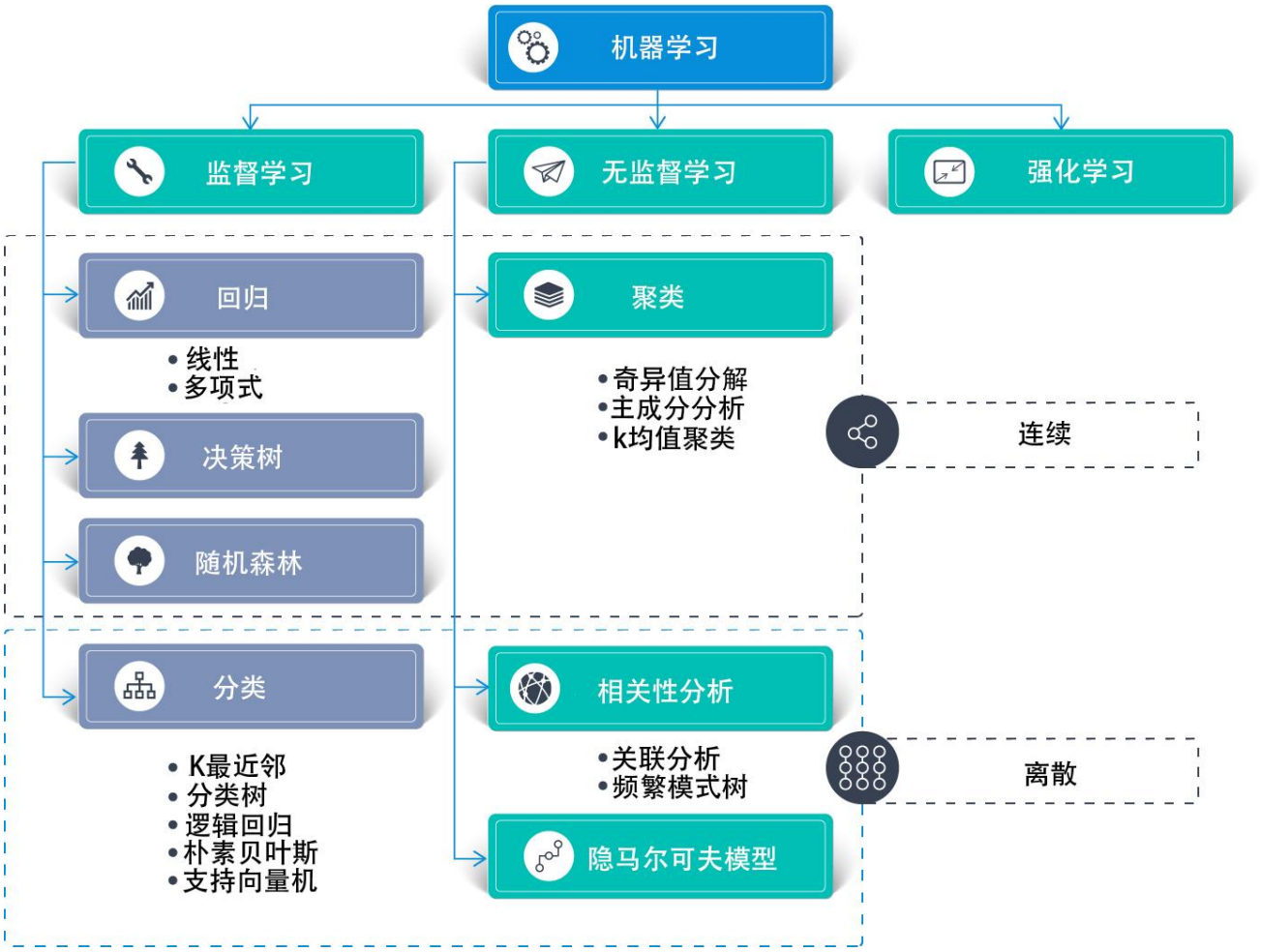

图 9　各种机器学习算法

我使用了 sklearn 中提供的各种机器学习模型，在训练集上训练，通过测试集得出准确度，来进行比较。请见表 2。实现代码请见附录。

决策树是一种主要用于分类问题的有监督学习算法。令人惊讶的是，它既适用于分类变量，也适用于连续因变量。在该算法中，我们将种群分成两个或多个齐次集。这是基于最重要的属性/自变量来完成的，以便尽可能地创建不同的组。

随机森林是决策树集合的代名词。在随机森林中，我们收集了决策树（即所谓的“森林”）。为了根据属性对一个新对象进行分类，每棵树都给出一个分类，我们说树为这个类“投票”。森林选择投票最多的分类（超过森林中所有的树）。

逻辑回归是一种分类而不是回归算法。它用于根据给定的自变量集估计离散值（二进制值，如 0/1、是/否、真/假）。简单地说，它通过将数据拟合到 logit 函数来预测事件发生的概率。因此，它也被称为 logit 回归。因为它预测了概率，所以它的输出值介于 0 和 1 之间（如预期）。

K 最近邻可以用于分类和回归问题。然而，它在工业分类问题中的应用更为广泛。K 近邻算法是一种简单的

算法，它存储所有可用的案例，并根据其 K 近邻的多数票对新案例进行分类。被分配给这个类的情况在它的 K 个最近邻中最为常见，这些最近邻是由距离函数度量的。

朴素贝叶斯是一种基于 Bayes 定理的分类技术，假设预测因子之间独立。简单地说，朴素贝叶斯分类器假设类中某个特定特征的存在与任何其他特征的存在无关。例如，如果一个水果是红色的，圆形的，直径大约 3 英寸，那么它可以被认为是一个苹果。即使这些特征彼此依赖或存在其他特征，朴素贝叶斯分类器将考虑所有这些属性来独立地贡献这种水果是苹果的概率。

支持向量机是一种分类方法。在该算法中，我们将每个数据项绘制为 n 维空间中的一个点（其中 n 是您拥有的特征数），每个特征的值是特定坐标的值。例如，如果我们只有两个特征，比如一个人的身高和头发长度，我们会首先在二维空间绘制这两个变量，其中每个点有两个坐标（这些坐标称为支持向量）。

梯度提升决策树是一种在处理大量数据进行高预测能力预测时使用的提升算法。提升算法实际上是一个学习算法的集合，它结合了多个基估计量的预测，以提高相对于单个估计量的稳健性。它将多个弱或平均预测值组合成一个强预测值。

模型	预测准确度
决策树	0.7623318385650224
随机森林	0.7668161434977578
逻辑回归	0.7847533632286996
K 最近邻	0.6995515695067265
朴素贝叶斯	0.7443946188340808
支持向量机	0.7443946188340808
梯度提升决策树	0.852017937219731

表 2　机器学习模型以及对应的准确率

由上表我们可以看到，相较于传统方法，梯度提升决策树的准确度最高，这是因为梯度提升决策树是一种迭代的决策树算法，由多棵决策树组成，所有树的结论累加起来作为最终答案。梯度提升决策树属于集成学习，集成学习不是单独的机器学习方法，而是通过构建并结合多个机器学习器来完成任务，集成学习可以用于分类问题集成、回归问题集成、特征选取集成、异常点检测集成等方面。其思想是对于训练数据集，我们通过训练若干个个体学习器，通过一定的结合策略形成一个强学习器，以达到博采众长的目的。

六、实现代码

```
# 导入相关库
import numpy as np
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy as sp
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn import metrics
from sklearn import model_selection
import pylab as pl
from sklearn.metrics import roc_curve

# 加载数据
datatrain =  pd.read_csv('../input/titanic/train.csv')
datatest =  pd.read_csv('../input/titanic/test.csv')

# 查看数据
print(datatest.head())
print(datatrain.info())
print('训练集列中为空值的记录: {} \n' .format( datatrain.isnull().sum()))
print('测试集列中为空值的记录: {}'.format( datatest.isnull().sum()))

# 清洗数据
datatrain['Age'].fillna(datatrain['Age'].median(), inplace = True)
datatrain['Embarked'].fillna(datatrain['Embarked'].mode()[0], inplace = True)
datatrain['Fare'].fillna(datatrain['Fare'].median(), inplace = True)
datatest['Age'].fillna(datatest['Age'].median(), inplace = True)
datatest['Embarked'].fillna(datatest['Embarked'].mode()[0], inplace = True)
datatest['Fare'].fillna(datatest['Fare'].median(), inplace = True)
drop_column = ['PassengerId','Cabin', 'Ticket']
datatrain.drop(drop_column, axis=1, inplace = True)
datatest.drop(drop_column, axis=1, inplace = True)
```

```
# 特征工程
alltables = [datatrain, datatest]
for dataset in alltables:
    dataset['FamilySize'] = dataset ['SibSp'] + dataset['Parch'] + 1
    dataset['IsAlone'] = 1
    dataset['IsAlone'].loc[dataset['FamilySize'] > 1] = 0
    dataset['Title'] = dataset['Name'].str.split(", ", expand=True)[1].str.split(".", expand=True)
[0]
    dataset['FareBin'] = pd.qcut(dataset['Fare'], 4)
    dataset['AgeBin'] = pd.cut(dataset['Age'].astype(int), 5)
stat_min = 10
title_names = (datatrain['Title'].value_counts() < stat_min)
datatrain['Title'] = datatrain['Title'].apply(lambda x: 'Misc' if title_names.loc[x] == True else
x)
print(datatrain['Title'].value_counts())

# 绘制图 4
sns.countplot(x="Survived", data=datatrain)

# 绘制图 5
fig, saxis = plt.subplots(2, 2,figsize=(16,12))
sns.countplot(x='Survived', hue="Embarked", data=datatrain,ax = saxis[0,0])
sns.countplot(x='Survived', hue="IsAlone", data=datatrain,ax = saxis[0,1])
sns.countplot(x="Survived", hue="Pclass", data=datatrain, ax = saxis[1,0])
sns.countplot(x="Survived", hue="Sex", data=datatrain, ax = saxis[1,1])

# 绘制图 6
fig, (axis1,axis2,axis3) = plt.subplots(1,3,figsize=(14,12))
sns.boxplot(x = 'Pclass', y = 'Fare', hue = 'Survived', data = datatrain, ax = axis1)
axis1.set_title('Pclass vs Fare Survival Comparison')
sns.violinplot(x = 'Pclass', y = 'Age', hue = 'Survived', data = datatrain, split = True, ax = axi
s2)
axis2.set_title('Pclass vs Age Survival Comparison')
sns.boxplot(x = 'Pclass', y ='FamilySize', hue = 'Survived', data = datatrain, ax = axis3)
```

```
axis3.set_title('Pclass vs Family Size Survival Comparison')

# 绘制图 7
a = sns.FacetGrid( datatrain, hue = 'Survived', aspect=4 )
a.map(sns.kdeplot, 'Age', shade= True )
a.set(xlim=(0 , datatrain['Age'].max()))
a.add_legend()

# 绘制图 8
plt.subplots(figsize =(14, 12))
correlation = datatrain.corr()
sns.heatmap(correlation, annot=True,cmap='coolwarm')

# 数据转换
label = LabelEncoder()
for dataset in alltables:
    dataset['Sex_Code'] = label.fit_transform(dataset['Sex'])
    dataset['Embarked_Code'] = label.fit_transform(dataset['Embarked'])
    dataset['Title_Code'] = label.fit_transform(dataset['Title'])
    dataset['AgeBin_Code'] = label.fit_transform(dataset['AgeBin'])
    dataset['FareBin_Code'] = label.fit_transform(dataset['FareBin'])
Target = ['Survived']
datatrain_x = ['Sex','Pclass', 'Embarked', 'Title','SibSp', 'Parch', 'Age', 'Fare', 'FamilySize',
'IsAlone']
datatrain_x_calc = ['Sex_Code','Pclass', 'Embarked_Code', 'Title_Code','SibSp', 'Parch', 'Age', 'F
are']
datatrain_xy =  Target + datatrain_x
print('Original X Y: ', datatrain_xy, '\n')
datatrain_x_bin = ['Sex_Code','Pclass', 'Embarked_Code', 'Title_Code', 'FamilySize', 'AgeBin_Code'
, 'FareBin_Code']
datatrain_xy_bin = Target + datatrain_x_bin
print('Bin X Y: ', datatrain_xy_bin, '\n')
datatrain_dummy = pd.get_dummies(datatrain[datatrain_x])
datatrain_x_dummy = datatrain_dummy.columns.tolist()
datatrain_xy_dummy = Target + datatrain_x_dummy
```

```
train1_x_dummy, test1_x_dummy, train1_y_dummy, test1_y_dummy = train_test_split(datatrain[datatrai
n_x_calc], datatrain[Target], random_state = 0)
train1_x_bin, test1_x_bin, train1_y_bin, test1_y_bin = model_selection.train_test_split(datatrain[
datatrain_x_bin], datatrain[Target] , random_state = 0)

# 决策树
from sklearn.tree import DecisionTreeClassifier
Model = DecisionTreeClassifier()
Model.fit(train1_x_dummy, train1_y_dummy)
y_predL = Model.predict(test1_x_dummy)
print(classification_report(test1_y_dummy, y_predL))
print(confusion_matrix(test1_y_dummy, y_predL))
print('准确率是',accuracy_score(y_predL,test1_y_dummy))

# 随机森林
from sklearn.ensemble import RandomForestClassifier
Model=RandomForestClassifier(max_depth=2)
Model.fit(train1_x_dummy, train1_y_dummy)
y_predR=Model.predict(test1_x_dummy)
print(classification_report(test1_y_dummy,y_predR))
print(confusion_matrix(y_predR,test1_y_dummy))
print('准确率是',accuracy_score(y_predR,test1_y_dummy))

# 逻辑回归
from sklearn.linear_model import LogisticRegression
Model = LogisticRegression()
Model.fit(train1_x_dummy, train1_y_dummy)
y_predLR = Model.predict(test1_x_dummy)
print(classification_report(test1_y_dummy, y_predLR))
print(confusion_matrix(test1_y_dummy, y_predLR))
print('准确率是',accuracy_score(y_predLR,test1_y_dummy))

# K 最近邻
from sklearn.neighbors import KNeighborsClassifier
Model = KNeighborsClassifier(n_neighbors=8)
```

```
Model.fit(train1_x_dummy, train1_y_dummy)
y_predKN = Model.predict(test1_x_dummy)
print(classification_report(test1_y_dummy, y_predKN))
print(confusion_matrix(test1_y_dummy, y_predKN))
print('准确率是',accuracy_score(y_predKN,test1_y_dummy))

# 朴素贝叶斯
from sklearn.naive_bayes import GaussianNB
Model = GaussianNB()
Model.fit(train1_x_dummy, train1_y_dummy)
y_predN = Model.predict(test1_x_dummy)
print(classification_report(test1_y_dummy, y_predN))
print(confusion_matrix(test1_y_dummy, y_predN))
print('准确率是',accuracy_score(y_predN,test1_y_dummy))

# 支持向量机
from sklearn.svm import SVC
Model = SVC()
Model.fit(train1_x_dummy, train1_y_dummy)
y_predSVM = Model.predict(test1_x_dummy)
print(classification_report(test1_y_dummy, y_predSVM))
print(confusion_matrix(test1_y_dummy, y_predSVM))
print('准确率是',accuracy_score(y_predSVM,test1_y_dummy))

# 梯度提升决策树
from sklearn.ensemble import GradientBoostingClassifier
ModelG=GradientBoostingClassifier()
ModelG.fit(train1_x_dummy, train1_y_dummy)
y_predGR=ModelG.predict(test1_x_dummy)
print(classification_report(test1_y_dummy,y_predGR))
print(confusion_matrix(y_predGR,test1_y_dummy))
print('准确率是',accuracy_score(y_predGR,test1_y_dummy))
```

七、参考文献

[1] 人工智能.机械工业出版社.史忠植.2018.1

[2] Titanic: Machine Learning from Disaster Data Description. https://www.kaggle.com/c/titanic/data

[3] scikit-learn: machine learning in Python. https://scikit-learn.org/stable/user_guide.html

[4] seaborn: statistical data visualization. http://seaborn.pydata.org/tutorial.html