# 决策树模型在泰坦尼克乘客生存预测中应用

学号: 18401190103 姓名： 谢威

学号: 18401190202 姓名： 陆镇涛

学号: 18401190120 姓名： 曹鹏霄

学号: 18401190107 姓名： 王云浩 提交时间： 2020-10-21

## 一、实验目的

1．了解决策树相关概念；

2．理解决策树构造过程；

3．应用决策树解决实际问题

4．学会应用决策树可视化工具；

## 实验内容

运用决策树算法给出泰坦尼克号存活预测。

## 实现过程分析

1. **数据处理与分析：**

首先通过matplotlib画图工具对数据进行初步的可视化分析

1. train = pd.read\_csv("train.csv") #读取文件
2. test = pd.read\_csv("test.csv")
3. fig = plt.figure()
4. fig.set(alpha=0.2)  # 设定图表颜色alpha参数
5. plt.rcParams['font.sans-serif'] = ['SimHei']
6. plt.rcParams['axes.unicode\_minus'] = False
7. plt.subplot2grid((2, 3), (0, 0))  # 在一张大图里分列几个小图
8. train.Survived.value\_counts().plot(kind='bar')  # 柱状图
9. plt.title(u"获救情况 (1为获救)")  # 标题
10. plt.ylabel(u"人数")
12. plt.subplot2grid((2, 3), (0, 1))
13. train.Pclass.value\_counts().plot(kind="bar")
14. plt.ylabel(u"人数")
15. plt.title(u"乘客等级分布")
17. plt.subplot2grid((2, 3), (0, 2))
18. plt.scatter(train.Survived, train.Age)
19. plt.ylabel(u"年龄")  # 设定纵坐标名称
20. plt.grid(b=True, which='major', axis='y')
21. plt.title(u"按年龄看获救分布 (1为获救)")
23. plt.subplot2grid((2, 3), (1, 0), colspan=2)
24. train.Age[train.Pclass == 1].plot(kind='kde')
25. train.Age[train.Pclass == 2].plot(kind='kde')
26. train.Age[train.Pclass == 3].plot(kind='kde')
27. plt.xlabel(u"年龄")  # plots an axis lable
28. plt.ylabel(u"密度")
29. plt.title(u"各等级的乘客年龄分布")
30. plt.legend((u'头等舱', u'2等舱', u'3等舱'), loc='best')  # sets our legend for our graph.
32. plt.subplot2grid((2, 3), (1, 2))
33. train.Embarked.value\_counts().plot(kind='bar')
34. plt.title(u"各登船口岸上船人数")
35. plt.ylabel(u"人数")
36. plt.show()

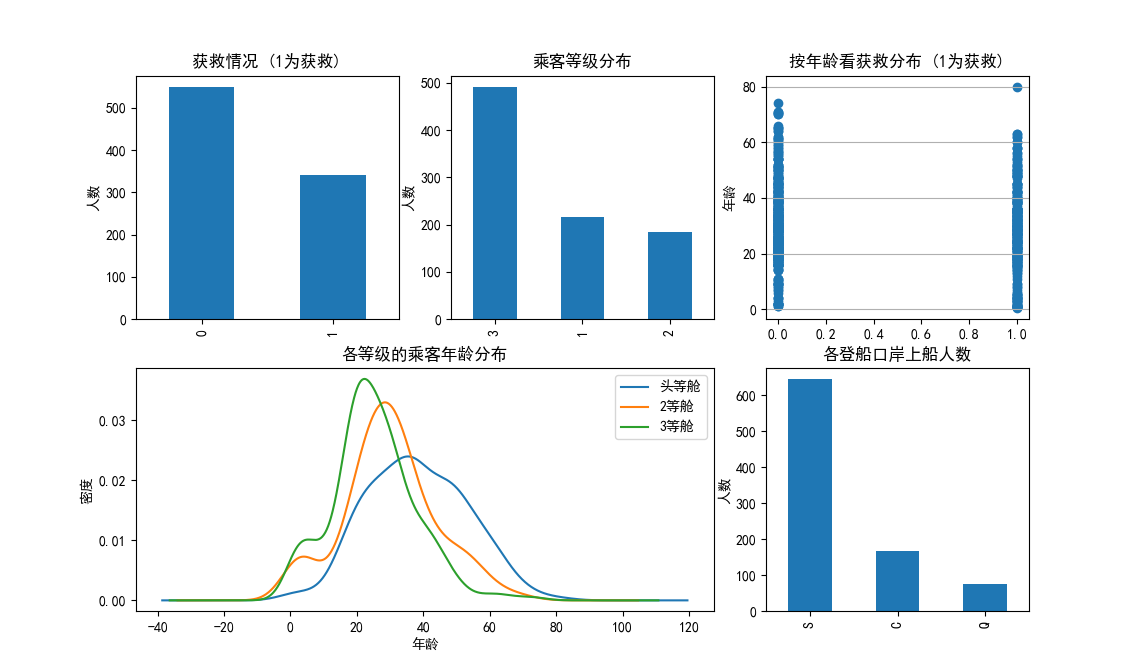


图 1

由于Age列有大量的数值缺失，所以需要对其进行数据填充，否则处理数据时会出错

1. train["Age"] = train["Age"].fillna(np.mean(train["Age"]))
2. test["Age"] = test["Age"].fillna(np.mean(test["Age"]))

通过分析列属性，发现NAME，Cabin，Ticket，Embarked对建立模型存在干扰或者并没 有对预测结果有影响，所以除去这几个属性。

1. train.drop("Cabin",axis=1,inplace=True)
2. test.drop("Cabin",axis=1,inplace=True)
3. train["Age"].isnull().sum()
4. test["Age"].isnull().sum()
5. train.drop("Name",axis=1,inplace=True)
6. test.drop("Name",axis=1,inplace=True)
7. train.drop("Ticket",axis=1,inplace=True)
8. test.drop("Ticket",axis=1,inplace=True)
9. train.drop("Embarked",axis=1,inplace=True)
10. test.drop("Embarked",axis=1,inplace=True)

由于Sex列是字符型的数据，我们需要把它转化成数值类型的数据方便进行处理，否则 在数据进行时会发生数据类型错误。

1. lab = LabelEncoder()
2. train["Sex"] = lab.fit\_transform(train["Sex"])#对sex列进行数值化
3. test["Sex"] = lab.fit\_transform(test["Sex"])

对Age和Fare进行归一化，数值进行放大缩小，方便计算

1. minmax = MinMaxScaler()#实例化函数
2. train["Age"] = minmax.fit\_transform(np.array(train["Age"]).reshape(-1,1))
3. test["Age"] = minmax.fit\_transform(np.array(test["Age"]).reshape(-1,1))
4. train["Age"].min()
5. train.info()
6. test.head()
7. minmax = MinMaxScaler()
8. train["Fare"] = minmax.fit\_transform(np.array(train["Fare"]).reshape(-1,1))

删除训练集里的Survived列，将Survived作为训练集的标签。

1. x = train.drop("Survived",axis=1)
2. x.info()
3. y = train["Survived"]

按比例分割训练集和测试集，这里我们将测试数据占样本比设为了0.2，保证训练样本 足以让训练出来的模型更加精确。

1. x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.2,random\_state=0)
2. **模型建立：**

通过sklearn中的DecisionTreeClassifier函数来建立存活情况(Survived)分别与其他几个属性的决策树模型,这里我们的特征选择标准criterion采用默认的gini，即CRAT，分类树回归算法。理由是这种算法很擅长处理计算量要求不大的连续种类的字段，并且生成的决策树可以清晰的反映出哪些字段比较重要。

1. lin = DecisionTreeClassifier(criterion='gini')
2. lin.fit(x\_train,y\_train)
3. y\_pred = lin.predict(x\_test)
4. mae = mean\_absolute\_error(y\_pred,y\_test)
5. predict = lin.predict(test)
6. **决策树可视化：**

通过graphviz图形可视化工具对训练所得的决策树进行绘制

1. dot\_data = tree.export\_graphviz(lin, out\_file=None)
2. graph = graphviz.Source(dot\_data)
3. graph.view()

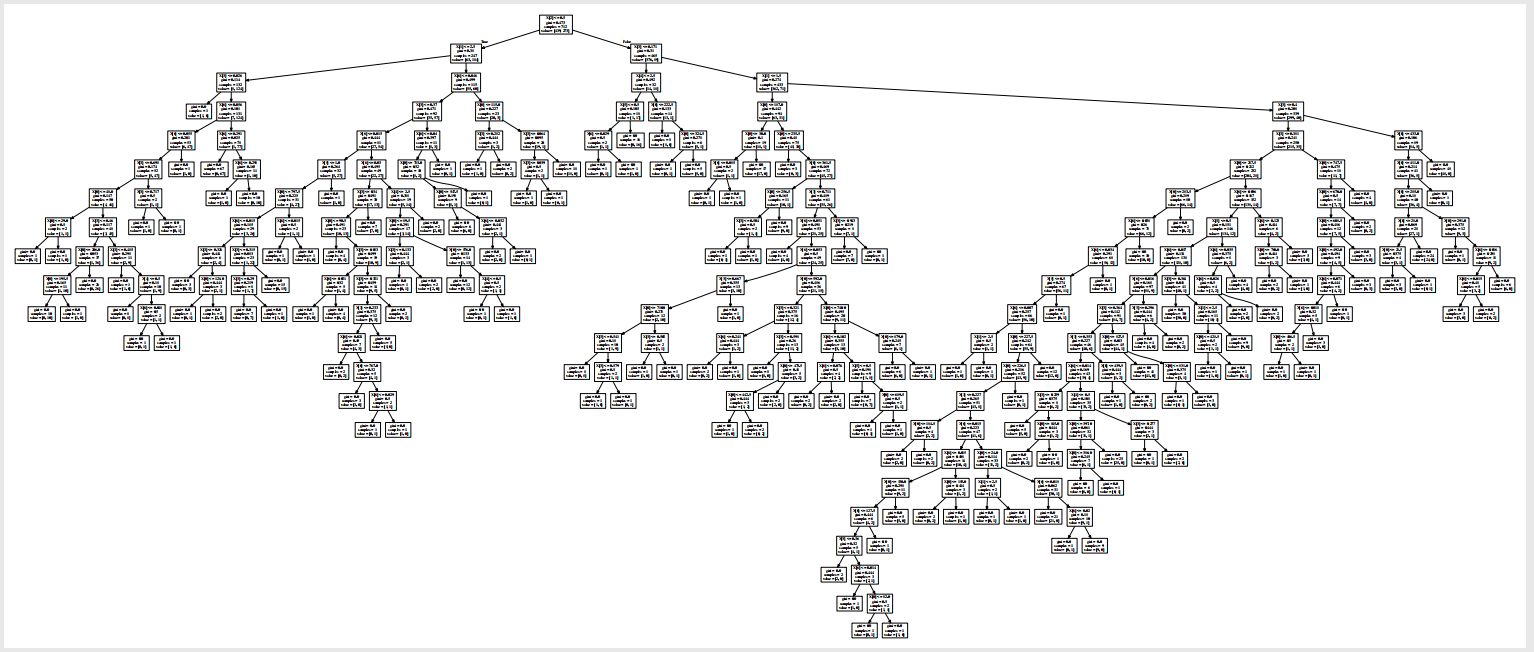


图 2

1. **实验结果：**

通过训练出的结果，我们可以初步得到其准确率

1. train\_score = lin.score(x\_train,y\_train)
2. test\_score = lin.score(x\_test,y\_test)
3. **print**('train score:{0}; test score:{1}'.format(train\_score,test\_score))



从这里可以看到训练分数非常高：100%，而测试分数只有75.9%，说明模型过拟合， 需要进行剪枝。

1. **参数优化：**

可以用max\_depth来控制决策树的深度，当决策树达到限定深度的时候，就不再进行 分裂。

1. **def** cv\_score(d):
2. clf = DecisionTreeClassifier(max\_depth=d)
3. clf.fit(x\_train, y\_train)
4. tr\_score = clf.score(x\_train, y\_train)
5. cv\_score = clf.score(x\_test, y\_test)
6. **return** (tr\_score, cv\_score)

9. depths = range(2, 15)
10. scores = [cv\_score(d) **for** d **in** depths]
11. tr\_scores = [s[0] **for** s **in** scores]
12. cv\_scores = [s[1] **for** s **in** scores]
14. # 找出交叉验证数据集评分最高的索引
15. best\_score\_index = np.argmax(cv\_scores)
16. best\_score = cv\_scores[best\_score\_index]
17. best\_param = depths[best\_score\_index]
18. **print**('best param:{0};best score:{1}'.format(best\_param, best\_score))



可以看到过拟合现象得到了减弱

## 结论

1. 根据实验分析，人物的性别（Sex），年龄（Age）与存活率（Survived）密切相关，可以看出女性的存活率要远大于男性,而婴儿（Age<4）的存活率往往比较高，大多数年龄处在15-25岁的乘客没有存活下来。
2. 当训练集准确率很高，测试集很低的时候，就代表模型出现了过拟合，这时候就需要使用剪枝操作，用max\_depth来控制决策树的深度，当决策树达到限定深度的时候，就不再进行分裂。

## 心得体会

通过学习本篇泰坦尼克号的分析实践，从中有很多的收获，首先给我提供了一种推测并补全缺失数据的空值处理方式，让我初步了解了决策树的（机器学习）分析方法，并使用决策树算法建立模型等。我们也初步明白了怎么对过拟合现想进行处理，怎么对决策树进行剪枝操作，这为我日后继续学习数据分析打下坚实的基础。