**数据挖掘实验报告**

**题目 泰坦尼克获救预测**

**姓名 郑阳**

**学号 1811082065**

# 题目背景及要求

泰坦尼克号的沉没是历史上最严重的沉船事故之一。 1912年4月15日，在她的处女航中，泰坦尼克号在与冰山相撞后沉没，在2224名乘客和机组人员中造成1502人死亡。这场耸人听闻的悲剧震惊了国际社会。造成海难失事的原因之一是乘客和机组人员没有足够的救生艇。尽管幸存下沉有一些运气因素，但有些人比其他人更容易生存，比如妇女，儿童和上流社会。

要求完成对哪些人可能存活的分析。

提供两份数据，训练数据train.csv,测试数据test.csv。最终预测test.csv中乘客的获救情况。

# 做题流程

1. 引用的库

2. 读取数据集并大概分析数据

3. 处理缺失值

4. 数据可视化分析

5. 特征编码与融合

6. 特征标准化

7. 建模

8. 模型评估与选择

1. **引用库**

***Numpy, pandas, seaborn, pyplot***



1. **读取数据集并大概分析数据**

***training = pd.read\_csv("train.csv")***

***testing = pd.read\_csv("test.csv")***

***training.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 |

**有以下这些字段**

PassengerId => 乘客ID  
Pclass => 乘客等级(1/2/3等舱位)  
Name => 乘客姓名  
Sex => 性别  
Age => 年龄  
SibSp => 堂兄弟/妹个数  
Parch => 父母与小孩个数  
Ticket => 船票信息  
Fare => 票价  
Cabin => 客舱  
Embarked => 登船港口

**2.1查看数据情况**

***training.info()***

<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 714 non-null float64

SibSp 891 non-null int64

Parch 891 non-null int64

Ticket 891 non-null object

Fare 891 non-null float64

Cabin 204 non-null object

Embarked 889 non-null object

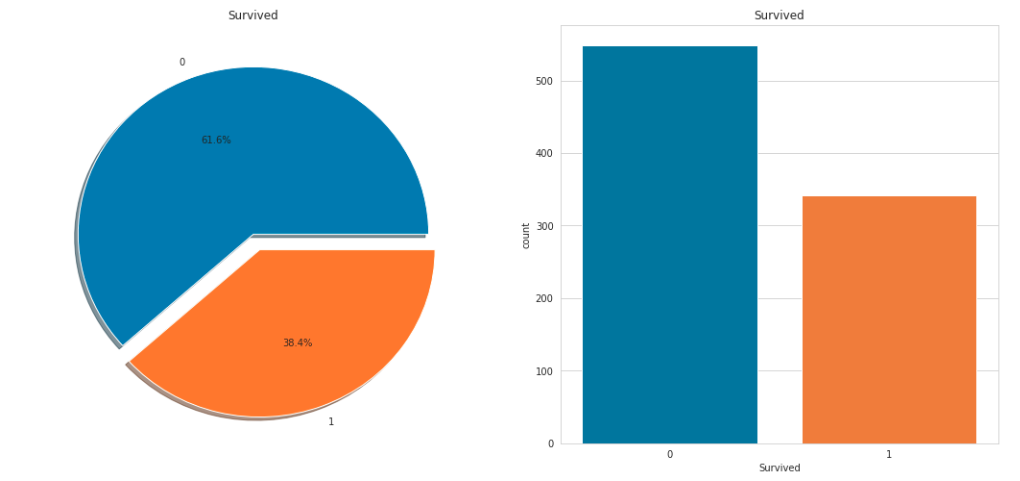
dtypes: float64(2), int64(5), object(5)

memory usage: 83.6+ KB

训练数据中总共有891名乘客，但属性的数据不全

* Age（年龄）只有714名乘客有记录
* Cabin（客舱）只有204名乘客有记录

同时发现只有只有年龄与船票价格为连续值，而性别（男，女） 登船地点（S,Q,C）为离散值。



通过画出获救图可以看出大部分人没有生还，生还者 342 人, 549 人未生还

1. **处理缺失值**

对于特征缺失值的处理有4种方法。

* 方法1：直接使用均值/中位数进行填充
* 方法2：根据经验值填充
* 方法3：回归模型预测
* 方法4：剔除掉

Note:

* 1. 如果缺值的样本占总数比例极高，可以直接舍弃
  2. 如果缺值的样本适中，而该属性非连续值特征属性(比如说类目属性)，那就把NaN作为一个新类别，加到类别特征中
  3. 如果缺值的样本适中，而该属性为连续值特征属性，有时候我们会考虑给定一个step，然后把它离散化，之后把NaN作为一个type加到属性类目中。
  4. 有些情况下，缺失的值个数并不是特别多，也可以试着根据已有的值，拟合一下数据，补充上。
  5. **首先查看缺失值的特征**

Training Data Missing

PassengerId 0

Survived 0

Pclass 0

Name 0

Sex 0

Age 177

SibSp 0

Parch 0

Ticket 0

Fare 0

Cabin 687

Embarked 2

dtype: int64

Testing Data Missing

PassengerId 0

Pclass 0

Name 0

Sex 0

Age 86

SibSp 0

Parch 0

Ticket 0

Fare 1

Cabin 327

Embarked 0

dtype: int64

**发现缺失的特征为Age（年龄）,Cabin（客舱号）,Embarked（登陆点）**

* + 1. **对于Cabin缺失过多，先分析统计情况**

***training.Cabin.value\_counts()***

C23 C25 C27 4

G6 4

B96 B98 4

D 3

C22 C26 3

E101 3

F2 3

F33 3

B57 B59 B63 B66 2

C68 2

B58 B60 2

E121 2

D20 2

E8 2

E44 2

B77 2

C65 2

D26 2

E24 2

E25 2

B20 2

C93 2

发现Cabin的分布过于离散，且因为缺失量过大，所以连同无关变量Ticket（船票编号）一同删掉。

* + 1. **对于Embarked，因为只缺失了2个，观察其分布，用众数进行填充**

***training.Embarked.value\_counts()***

S 644

C 168

Q 77

Name: Embarked, dtype: int64

***training['Embarked'].fillna('S', inplace=True)***

***null\_value(training, testing)***

* + 1. **对于Fare，只在测试集缺失一条，直接用中位数填充**

***testing['Fare'].fillna(testing['Fare'].median(), inplace=True)***

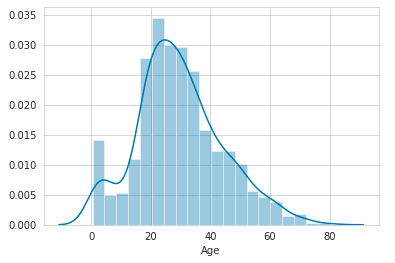
***null\_value(training, testing)***

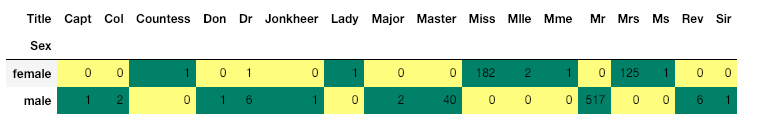
* + 1. **对于Age，首先观察其分布情况**

***copy = training.copy()***

***copy.dropna(inplace=True)***

***sns.distplot(copy.Age)***



因Age为连续值，可尝试分为多个step，来将其离散化。通过检查名字特征可以看到名字有像先生或夫人这样的称呼，这样就可以把先生和夫人的平均值分配给各自的组。

将所有的称呼替换为Master,Miss,Mr,Mrs,Other五种，再取均值对每个称呼的Age重新赋值。

***training.groupby('Title')['Age'].mean()***

Title

Master 4.574167

Miss 21.860000

Mr 32.739609

Mrs 35.981818

Other 45.888889

***def set\_missing\_age(data):***

***# 使用每组的均值来进行填充***

***data.loc[(data.Age.isnull()) & (data.Title=='Master'), 'Age'] = 5***

***data.loc[(data.Age.isnull()) & (data.Title=='Miss'), 'Age'] = 22***

***data.loc[(data.Age.isnull()) & (data.Title=='Mr'), 'Age'] = 33***

***data.loc[(data.Age.isnull()) & (data.Title=='Mrs'), 'Age'] = 36***

***data.loc[(data.Age.isnull()) & (data.Title=='Other'), 'Age'] = 46***

***set\_missing\_age(training)***

***set\_missing\_age(testing)***

最终缺失值处理完毕，所有缺失值都已填充完毕。

Training Data Missing

PassengerId 0

Survived 0

Pclass 0

Sex 0

Age 0

SibSp 0

Parch 0

Fare 0

Embarked 0

Title 0

dtype: int64

Testing Data Missing

PassengerId 0

Pclass 0

Sex 0

Age 0

SibSp 0

Parch 0

Fare 0

Embarked 0

Title 0

dtype: int64

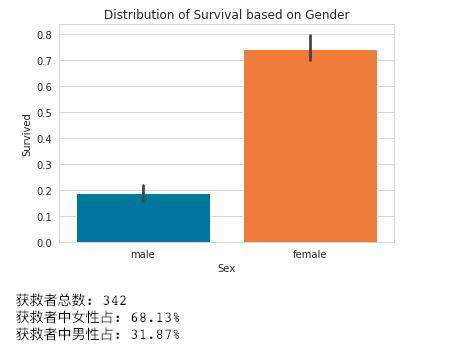
1. **数据可视化分析**

对于特征理解分析分为三部分

1. 单特征分析，逐个变量分析其对结果的影响
2. 多变量统计分析，综合考虑多种情况影响
3. 统计绘图得出结论

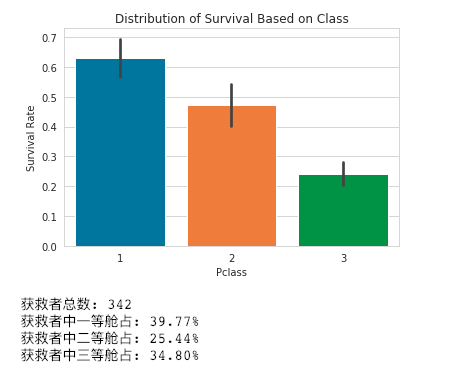
通过可视化分析从而得到特征的熵和信息益增

* 1. **性别特征**

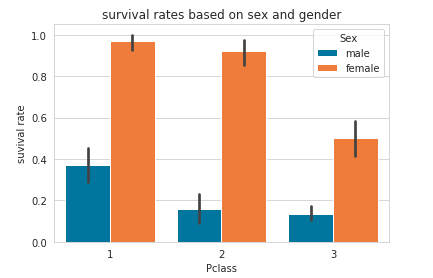


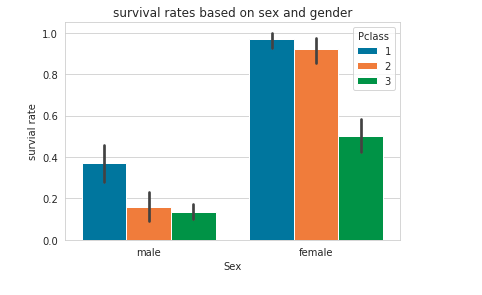
由上可得出虽然船上男性较多，但获救者中，女性是男性的2倍，而图中可得在船上，女性的生存几率是男性的三倍，大于70%。 可见，性别对于生存结果是个很重要的特征。

* 1. **船舱等级Pclass**



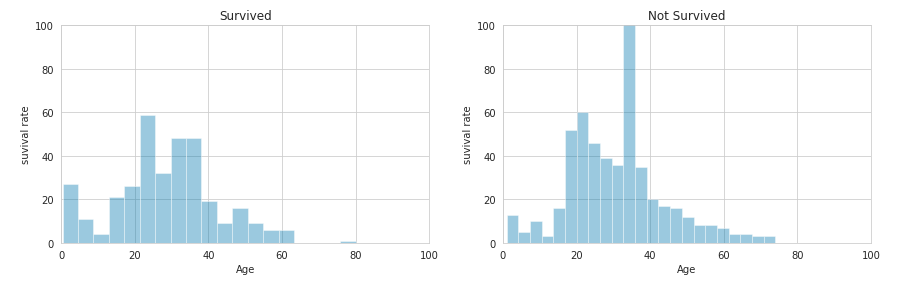
* 1. **性别与船舱等级对生存率的联合影响**

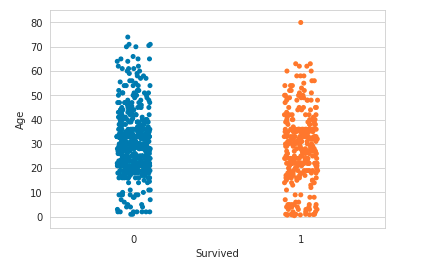
****



显然对于船舱等级而言，等级一的更容易存活。

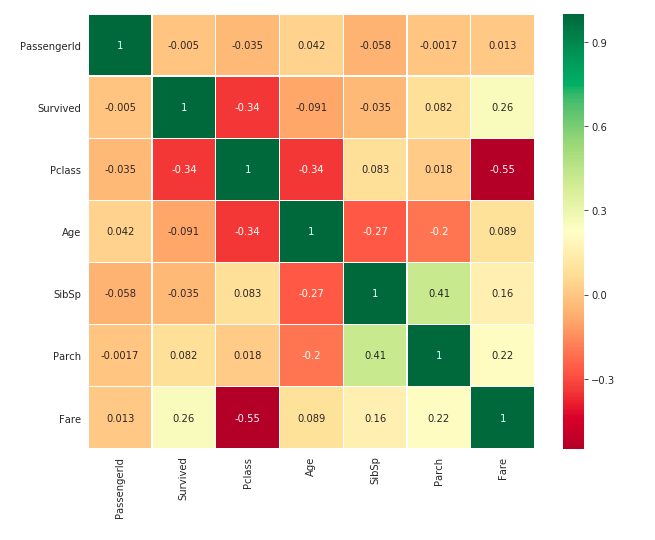
* 1. **年龄特征**





根据图表可以看出  
1）幼儿的获救率较高  
2）死亡人数最高的是30-40岁年龄组

* 1. **所有特征相关性热力图**



1. **特征编码与融合**

|  | **PassengerId** | **Survived** | **Pclass** | **Sex** | **Age** | **SibSp** | **Parch** | **Fare** | **Embarked** | **Title** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Mr |
| **1** | 2 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | Mrs |
| **2** | 3 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Miss |
| **3** | 4 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | Mrs |
| **4** | 5 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Mr |

现观察数据，发现部分特征如Sex，Embarked，Title仍旧为字符串，所以可以通过顺序编码或One-hot编码将其变为整型特征。具体可通过sklearn.preprocessing中的LabelEncoder方法fit进特征数据，再将要编码的特征transform即可。此处仅举例Sex编码代码。

**5.1 特征编码**

* + 1. **Sex性别编码**

***In: from sklearn.preprocessing import LabelEncoder***

***le\_sex = LabelEncoder()***

***le\_sex.fit(training['Sex'])***

***print list(le\_sex.classes\_)***

***encoded\_sex\_training = le\_sex.transform(training['Sex'])***

***training['Sex'] = encoded\_sex\_training***

***encoded\_sex\_testing = le\_sex.transform(testing['Sex'])***

***testing['Sex'] = encoded\_sex\_testing***

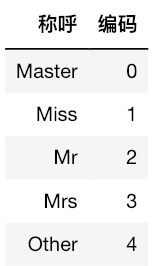
***Out: ['female', 'male']***

即将Sex的female，male编码为0， 1

* + 1. **Embarked编码**

C:0  
Q:1  
S:2

* + 1. **Title编码**

****

**最终得到处理后数据**

|  | **PassengerId** | **Survived** | **Pclass** | **Sex** | **Age** | **SibSp** | **Parch** | **Fare** | **Embarked** | **Title** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 0 | 3 | 1 | 22.0 | 1 | 0 | 7.2500 | 2 | 2 |
| **1** | 2 | 1 | 1 | 0 | 38.0 | 1 | 0 | 71.2833 | 0 | 3 |
| **2** | 3 | 1 | 3 | 0 | 26.0 | 0 | 0 | 7.9250 | 2 | 1 |
| **3** | 4 | 1 | 1 | 0 | 35.0 | 1 | 0 | 53.1000 | 2 | 3 |
| **4** | 5 | 0 | 3 | 1 | 35.0 | 0 | 0 | 8.0500 | 2 | 2 |

**都已为数值变量**

**5.2特征融合**

SibSp兄妹个数与Parch父母孩子个数这两个特征可以整合为家庭成员个数。融合为新特征*IsAlone*

***training['FamSize'] = training['SibSp'] + training['Parch'] + 1***

***testing["FamSize"] = testing["SibSp"] + testing["Parch"] + 1***

***training['IsAlone'] = training.FamSize.apply(lambda x:1 if x == 1 else 0)***

***testing['IsAlone'] = testing.FamSize.apply(lambda x:1 if x == 1 else 0)***

所有的特征都是数值变量，对于age和fare要进行归一化

1. **特征标准化**

***from sklearn.preprocessing import StandardScaler***

***def value\_scaler(data):***

***scaler = StandardScaler()***

***ages = np.array(data['Age']).reshape(-1, 1)***

***fares = np.array(data['Fare']).reshape(-1, 1)***

***data['Age'] = scaler.fit\_transform(ages)***

***data['Fare'] = scaler.fit\_transform(fares)***

***value\_scaler(training)***

***value\_scaler(testing)***

归一化后数据预览：

|  | **PassengerId** | **Survived** | **Pclass** | **Sex** | **Age** | **SibSp** | **Parch** | **Fare** | **Embarked** | **Title** | **FamSize** | **IsAlone** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 0 | 3 | 1 | -0.590771 | 1 | 0 | -0.502445 | 2 | 2 | 2 | 0 |
| **1** | 2 | 1 | 1 | 0 | 0.614586 | 1 | 0 | 0.786845 | 0 | 3 | 2 | 0 |
| **2** | 3 | 1 | 3 | 0 | -0.289432 | 0 | 0 | -0.488854 | 2 | 1 | 1 | 1 |
| **3** | 4 | 1 | 1 | 0 | 0.388582 | 1 | 0 | 0.420730 | 2 | 3 | 2 | 0 |
| **4** | 5 | 0 | 3 | 1 | 0.388582 | 0 | 0 | -0.486337 | 2 | 2 | 1 | 1 |

1. **模型训练与预测**

**使用如下模型进行训练，并进行打分**

1) 支持向量机  
2) 随机森林  
3) 逻辑回归  
4) k-临近  
5) 朴素贝叶斯  
6) 决策树

**1. 将训练数据切分为训练集与验证集**

**2. 通过GridSearchCV来找出所选模型的最佳超参**

**3. 通过metrics库的make\_scorer与accuracy\_score来评价模型性能[¶](http://localhost:8888/notebooks/titanic/titanic.ipynb" \l "%E9%80%9A%E8%BF%87metrics%E5%BA%93%E7%9A%84make_scorer%E4%B8%8Eaccuracy_score%E6%9D%A5%E8%AF%84%E4%BB%B7%E6%A8%A1%E5%9E%8B%E6%80%A7%E8%83%BD)**

**本次实验用了SVM, 随机森林，逻辑回归与K邻近作为模型进行预测。举例随机森林代码如下。**

***rf\_clf = RandomForestClassifier()***

***#模型参数字典，以通过GridSearchCV来寻找最佳模型***

***parameters\_rf = {"n\_estimators": [4, 5, 6, 7, 8, 9, 10, 15], "criterion": ["gini", "entropy"], "max\_features": ["auto", "sqrt", "log2"], "max\_depth": [2, 3, 5, 10], "min\_samples\_split": [2, 3, 5, 10]}***

***grid\_rf = GridSearchCV(rf\_clf, parameters\_rf, scoring=make\_scorer(accuracy\_score))***

***grid\_rf.fit(X\_training, y\_training)***

***rf\_clf = grid\_rf.best\_estimator\_***

***rf\_clf.fit(X\_training, y\_training)***

***pred\_rf = rf\_clf.predict(X\_valid)***

***acc\_rf = accuracy\_score(y\_valid, pred\_rf)***

***print "RandomForest score: %s" % (str(acc\_rf))***

RandomForest score: 0.8044692737430168

1. **评估各模型性能并选择最优模型进行预测**

***In: model\_performance = pd.DataFrame({***

***"Model": ["SVC", "Random Forest", "Logistic Regression", "K Nearest Neighbors"],***

***"Accuracy": [acc\_svc, acc\_rf, acc\_logreg, acc\_knn]***

***})***

***model\_performance.sort\_values(by="Accuracy", ascending=False)***

***Out:***

|  | **Accuracy** | **Model** |
| --- | --- | --- |
| **0** | 0.821229 | SVC |
| **1** | 0.804469 | Random Forest |
| **3** | 0.804469 | K Nearest Neighbors |
| **2** | 0.798883 | Logistic Regression |

最终选择随机森林作为模型，用所有train数据作为训练数据，预测结果

***rf\_clf.fit(X\_train, y\_train)***

***result\_predictions = rf\_clf.predict(X\_test)***

***result = pd.DataFrame({***

***"PassengerId" : testing['PassengerId'],***

***"Survived": result\_predictions***

***})***

***result.to\_csv("titanic.csv", index=False)***

***print result***

PassengerId Survived

0 892 0

1 893 0

2 894 0

3 895 0

4 896 0

5 897 0

6 898 1

7 899 0

8 900 1

9 901 0

10 902 0

11 903 0

12 904 1

13 905 0

14 906 1

15 907 1

16 908 0

17 909 0

18 910 0

19 911 1

20 912 0

21 913 1

22 914 1

23 915 0

24 916 1

25 917 0

26 918 1

27 919 0

28 920 0

29 921 0

.. ... ...

388 1280 0

389 1281 0

390 1282 0

391 1283 1

392 1284 1

393 1285 0

394 1286 0

395 1287 1

396 1288 0

397 1289 1

398 1290 0

399 1291 0

400 1292 1

401 1293 0

402 1294 1

403 1295 0

404 1296 0

405 1297 0

406 1298 0

407 1299 0

408 1300 0

409 1301 0

410 1302 0

411 1303 1

412 1304 0

413 1305 0

414 1306 1

415 1307 0

416 1308 0

417 1309 1

[418 rows x 2 columns]