数据挖掘作业三

——分类与聚类

姓名： 高建花

班级： 硕士 4 班

学号： 2120171010

分类与聚类

1. 数据集

https://www.kaggle.com/c/titanic/data

此数据集包括训练数据(train.csv)和测试数据(test.csv)，训练数据用于建立机器学习的模型，测试集用来检测模型的有效性。

2. 编程环境

编程语言：python

Packages：sklearn, pandas, matplotlib

IDE：PyCharm

3. 分类与聚类

3.1 数据预处理

先查看训练数据和测试数据的基本信息

train.info	test.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: 66.2+ KB	<class 'pandas.core.frame.DataFrame'> RangeIndex: 418 entries, 0 to 417 Data columns (total 11 columns): PassengerId 418 non-null int64 Pclass 418 non-null int64 Name 418 non-null object Sex 418 non-null object Age 332 non-null float64 SibSp 418 non-null int64 Parch 418 non-null int64 Ticket 418 non-null object Fare 417 non-null float64 Cabin 91 non-null object Embarked 418 non-null object dtypes: float64(2), int64(4), object(5) memory usage: 27.8+ KB

考虑到 PassengerId、Name、Ticket 是标识属性，对于 Survived 的预测没有影响，故将这些属性丢弃；另外测试数据和训练数据中 Cabin 属性的缺失值均比较多，故将此属性也丢弃。然后使用均值来填补训练数据中的 Age 和测试数据中的 Age、Fare 属性，用众数来填补训练数据中的 Embarked 属性。得到的数据信息如下：

train.info	test.info
<class 'pandas.core.frame.DataFrame'> RangeIndex: 891 entries, 0 to 890 Data columns (total 7 columns): Pclass 891 non-null int64 Sex 891 non-null object Age 891 non-null float64 SibSp 891 non-null int64 Parch 891 non-null int64 Fare 891 non-null float64 Embarked 891 non-null object dtypes: float64(2), int64(3), object(2) memory usage: 41.8+ KB	<class 'pandas.core.frame.DataFrame'> RangeIndex: 418 entries, 0 to 417 Data columns (total 7 columns): Pclass 418 non-null int64 Sex 418 non-null object Age 418 non-null float64 SibSp 418 non-null int64 Parch 418 non-null int64 Fare 418 non-null float64 Embarked 418 non-null object dtypes: float64(2), int64(3), object(2) memory usage: 19.6+ KB

3.2 支持向量机分类

分类步骤如下：

(1) 设定初始参数值，比如核函数、正则化参数等；

(2) 使用网络搜索进行调参，核函数为 linear 时有正则化参数和权重向量，核函数为 rbf 时有 gamma、正则化参数和权重向量；

(3) 交叉验证。对于每组参数，使用 4 重交叉验证估计训练得到的分类器的泛化性能，即将训练样本分为 4 份，每次采用 3 份作为训练集，另 1 份作为验证集。重复做 4 次实验，每次采用不同的验证集。

(4) 将 4 次实验的评价指标求均值，作为此组参数获得的分类器其泛化性能的评价指标；

(5) 得到泛化性能最好的一组参数，在整个训练集上训练模型。

结果如下：

Train result
0.833 (+/-0.010) for {'C': 1, 'kernel': 'linear'}
0.762 (+/-0.054) for {'C': 1, 'kernel': 'rbf'}
0.832 (+/-0.012) for {'C': 10, 'kernel': 'linear'}
0.765 (+/-0.040) for {'C': 10, 'kernel': 'rbf'}
Best parameters:
{'C': 1, 'kernel': 'linear'}

因为本实验用的是普通的笔记本电脑，所以参数设置不宜过多，使用上述参数来简单展示。将最终的预测结果写入到文件，上传到 kaggle 网站上，查看得分如下：

Name	Submitted	Wait time	Execution time	Score
submission_SVM.csv	just now	0 seconds	0 seconds	0.76555

Complete

3.3 AdaBoost 分类

与上述 SVM 类似，不同的是 AdaBoost 中设置的参数不同，此处设置了弱分类器的个数 n_estimators 和学习率 learning_rate 的几组值：

```
parameters = {'n_estimators':[50, 100, 150, 200],
              'learning_rate':[1, 0.1, 0.01]}
```

训练结果如下：

Train result
Grid scores on validation set: 0.858 (+/-0.028) for {'learning_rate': 1, 'n_estimators': 50} 0.856 (+/-0.026) for {'learning_rate': 1, 'n_estimators': 100} 0.857 (+/-0.026) for {'learning_rate': 1, 'n_estimators': 150} 0.857 (+/-0.024) for {'learning_rate': 1, 'n_estimators': 200} 0.848 (+/-0.017) for {'learning_rate': 0.1, 'n_estimators': 50} 0.857 (+/-0.022) for {'learning_rate': 0.1, 'n_estimators': 100} 0.861 (+/-0.025) for {'learning_rate': 0.1, 'n_estimators': 150} 0.861 (+/-0.026) for {'learning_rate': 0.1, 'n_estimators': 200} 0.786 (+/-0.045) for {'learning_rate': 0.01, 'n_estimators': 50} 0.828 (+/-0.009) for {'learning_rate': 0.01, 'n_estimators': 100} 0.829 (+/-0.012) for {'learning_rate': 0.01, 'n_estimators': 150} 0.835 (+/-0.011) for {'learning_rate': 0.01, 'n_estimators': 200} Best parameters: {'learning_rate': 0.1, 'n_estimators': 200}

将预测结果上传到 kaggle 网站上，查看得分如下：

可看出，其分类的准确率比 SVM 稍低，当然还可以通过调整参数来提升其准确率。

3.4 KMeans 聚类

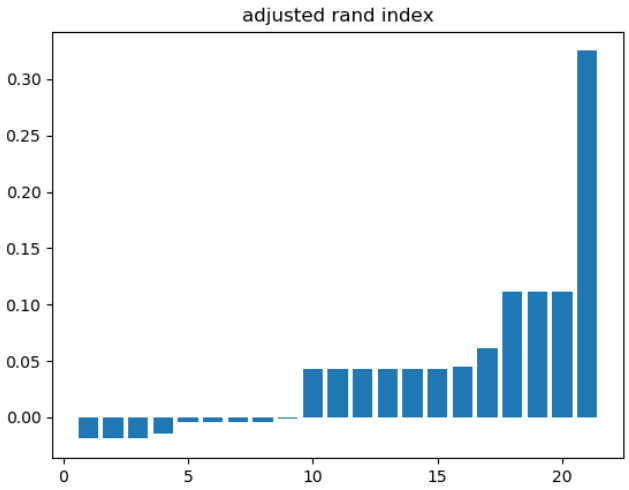
其主要步骤如下：

(1) 每次选取两个特征进行聚类；

(2) KMeans 中的聚类个数设置为 2，即 Survived 的值为 0 或 1。

(3) 使用 adjusted rand index(ARI，兰德指数)来作为聚类效果的评价指标，其取值范围为[-1,1]，负值表示聚类结果较差，越接近 1，其聚类结果越好。

最终得到结果如下图：可以看出(‘sex’，‘Embarked’)两个特征跟 Survived 的相关程度最高。

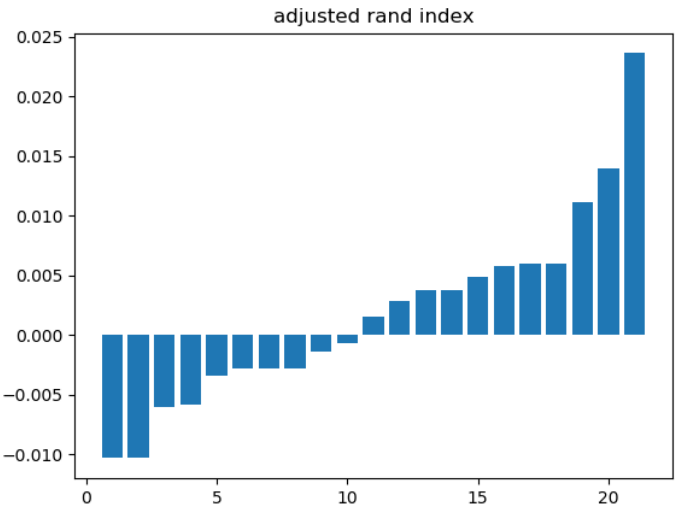



```
('SibSp', 'Parch')          : -0.020
('Pclass', 'SibSp')         : -0.019
('Sex', 'SibSp')            : -0.019
('SibSp', 'Embarked')       : -0.019
('Pclass', 'Age')           : -0.004
('Sex', 'Age')              : -0.004
('Age', 'SibSp')            : -0.004
('Age', 'Embarked')         : -0.004
('Age', 'Parch')            : -0.001
('Pclass', 'Fare')          : 0.043
('Sex', 'Fare')             : 0.043
('Age', 'Fare')             : 0.043
('SibSp', 'Fare')           : 0.043
('Parch', 'Fare')           : 0.043
('Fare', 'Embarked')        : 0.043
('Parch', 'Embarked')       : 0.045
('Sex', 'Parch')            : 0.061
('Pclass', 'Sex')           : 0.111
('Pclass', 'Parch')         : 0.111
('Pclass', 'Embarked')      : 0.111
('Sex', 'Embarked')         : 0.325
```

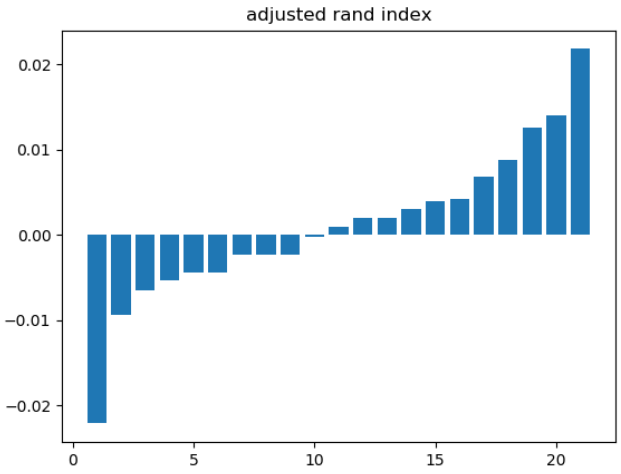
3.5 谱聚类

与 KMeans 聚类相类似，不同的是在谱聚类中，相似性度量分别使用 6-近邻和 8-近邻。得到结果如下图(依次为 6-近邻和 8-近邻的结果)：可以看出同等条件下，6-近邻比 8-近邻的聚类效果稍好，且与 Survived 相关程度最高的特征组合也不同，分别是('Pclass',' Fare'),('SibSp', 'Fare')。

两种聚类算法相比，可看出在一定的参数条件下，KMeans 的聚类效果要比谱聚类的效果好。




```
('SibSp', 'Fare')           : -0.010
('Sex', 'Age')              : -0.010
('Age', 'SibSp')            : -0.006
('Fare', 'Embarked')        : -0.006
('Pclass', 'Embarked')      : -0.003
('Pclass', 'Age')           : -0.003
('Sex', 'SibSp')            : -0.003
('Sex', 'Embarked')         : -0.003
('Parch', 'Fare')           : -0.001
('Pclass', 'Sex')           : -0.001
('Sex', 'Parch')            : 0.001
('Age', 'Embarked')         : 0.003
('Pclass', 'SibSp')         : 0.004
('Parch', 'Embarked')       : 0.004
('Sex', 'Fare')             : 0.005
('Age', 'Parch')            : 0.006
('Pclass', 'Parch')         : 0.006
('SibSp', 'Embarked')       : 0.006
('SibSp', 'Parch')          : 0.011
('Age', 'Fare')             : 0.014
('Pclass', 'Fare')          : 0.024
```




```
('Sex', 'Fare')             : -0.022
('Sex', 'Age')              : -0.009
('Sex', 'Parch')            : -0.006
('Pclass', 'Fare')          : -0.005
('Pclass', 'Embarked')      : -0.004
('SibSp', 'Embarked')       : -0.004
('Pclass', 'Sex')           : -0.002
('Age', 'SibSp')            : -0.002
('SibSp', 'Parch')          : -0.002
('Age', 'Parch')            : -0.000
('Age', 'Embarked')         : 0.001
('Parch', 'Fare')           : 0.002
('Parch', 'Embarked')       : 0.002
('Pclass', 'SibSp')         : 0.003
('Sex', 'SibSp')            : 0.004
('Sex', 'Embarked')         : 0.004
('Pclass', 'Age')           : 0.007
('Pclass', 'Parch')         : 0.009
('Fare', 'Embarked')        : 0.013
('Age', 'Fare')             : 0.014
('SibSp', 'Fare')           : 0.022
```