数据挖掘作业三报告

分类与聚类

姓名：赵赫　　学号：2120171103

一、 问题描述

本次作业中，我选择将对 Titanic 数据集（https://www.kaggle.com/c/titanic/data）进行分类与聚类。

本次分类与聚类任务包含以下四个子任务：

- 使用分类模型（至少 2 个）对数据集进行挖掘；
- 对挖掘结果进行可视化，并解释其意义；
- 使用聚类方法（至少 2 种）对数据集进行分析；
- 对挖掘结果进行可视化，并解释其意义。

二、 数据描述

- **数据集：Titanic**

  该数据集共包含 10 个属性特征，均为泰坦尼克号上乘客的相关信息，训练集包括 X 个乘客，测试集包括 X 个乘客。本任务需要通过训练集的乘客信息以及是否幸存的标签，预测测试集中乘客是否幸存。

三、 分类与聚类过程

3.1 数据预处理

为构造分类模型，首先通过分析训练数据集中的特征属性，我选择了 7 个特征，其中包括 Pclass、Sex、Age、Fare、Embarked、SibSp、Parch。此外，由于训练集和测试集中都存在缺失数据，因此需要对缺失数据进行填补，针对于数值属性，我们选择用中位数进行填补，针对于非数值属性，我们选择用频数最高值进行填补。填补所调用函数如图所示：

```
class DataFrameImputer(TransformerMixin):
    """We'll impute missing values using the median for numeric columns and the most
       common value for string columns.
       This is based on some nice code by 'sveitser' at http://stackoverflow.com/a/25562948"""
    def fit(self, X, y=None):
        self.fill = pd.Series([X[c].value_counts().index[0] if X[c].dtype == np.dtype('O')
                               else X[c].median() for c in X], index=X.columns)
        return self
    def transform(self, X, y=None):
        return X.fillna(self.fill)
```

在经过填补的数据基础上，再进行了特征工程，利用已有特征构造两个新特

征：Family 和 IsAlone。其中 Family 值为 SibSp 与 Parch 的加和，代表家人总数；而 IsAlone 值与 Family 值相关联，若 Family 为 0，则 IsAlone 为 1，否则 IsAlone 为 0，代表此乘客是否为独自一人乘船。对数据预处理的代码片段如下：

```
def load_data(self, path, train=True):
    # 读取数据
    data = pd.read_csv(path, header = 0)

    # 选定将使用的特征
    feature_columns_to_use = ['Pclass','Sex','Age','Fare','Embarked','SibSp','Parch']
    nonnumric_columns = ['Sex','Embarked']

    # 选取特征对应的数据，并对数据进行缺失值填补
    X = data[feature_columns_to_use]
    X_imputed = DataFrameImputer().fit_transform(X)

    # 进行特征工程，构造两个新特征
    X_imputed['Family'] = data['SibSp'] + data['Parch']
    X_imputed['IsAlone'] = pd.Series([1 if X_imputed['Family'][i]>0 else 0 for i in range(len(X_imputed))])

    # 将非数值属性转化为数值属性
    le = LabelEncoder()
    for feature in nonnumric_columns:
        X_imputed[feature] = le.fit_transform(X_imputed[feature])

    # 准备用于输入到模型中的数据
    input_X = X_imputed.as_matrix()
    if train == False:
        return input_X
    else:
        input_Y = data['Survived']
        return input_X, input_Y
```

3.2 分类过程

在经过预处理的数据集的基础上，分别采用 Xgboost、DecisionTree（决策树）、SVM（支持向量机）三种方法进行分类，并使用训练好的模型对测试集进行预测。构建分类器的代码片段如下：

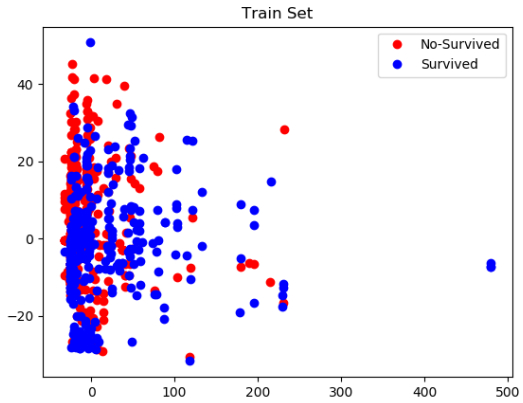
```
def classifier_xgboost(train_X, train_Y, test_X):
    gbm = xgb.XGBClassifier(max_depth=3, n_estimators=300, learning_rate=0.05)
    gbm.fit(train_X, train_Y)
    predictions = gbm.predict(test_X)
    return predictions

def classifier_dicisionTree(train_X, train_Y, test_X):
    dtc = DecisionTreeClassifier(random_state=0, criterion='gini', max_leaf_nodes=10)
    dtc.fit(train_X, train_Y)
    predictions = dtc.predict(test_X)
    return predictions

def classifier_SVM(train_X, train_Y, test_X):
    svc = svm.SVC()
    svc.fit(train_X, train_Y)
    predictions = svc.predict(test_X)
    return predictions
```

全部分类器模型的构建与预测位于“./src/classifiers”。

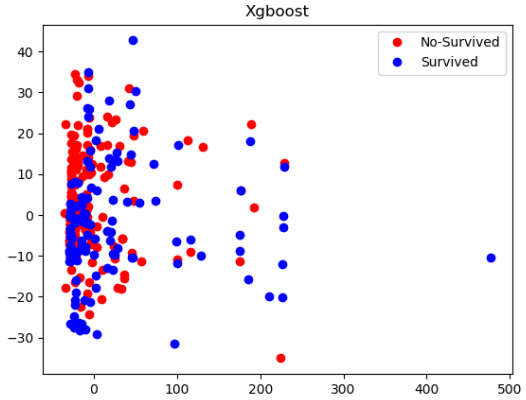
3.3 分类可视化结果

在进行可视化之前，为了将可视化高维特征数据的分布和分类结果，我们将多维的特征向量利用 PCA 降至 2 维。因此示意图上的 x 轴、y 轴分别表示降维后的两维数值，而红色点代表分类标签为 0（No-Survived），蓝色点代表分类标签为 1（Survived）。

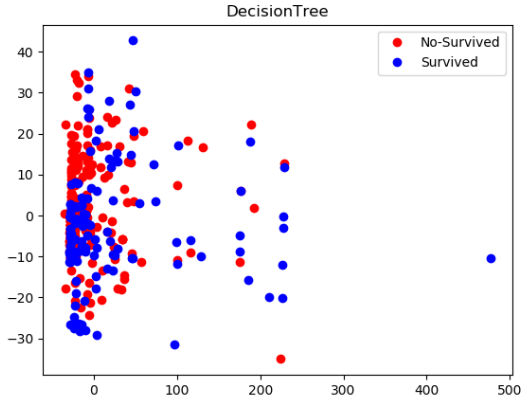
首先，我们可以看到在训练集上的数据分布如图所示：



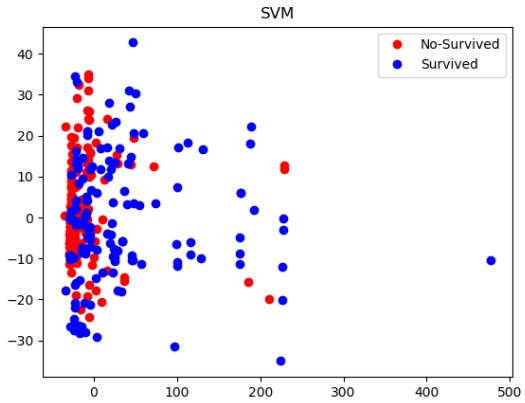
使用 Xgboost 分类器在测试集上的分类结果如图所示：



使用 DecisionTree 分类器在测试集上的分类结果如图所示：

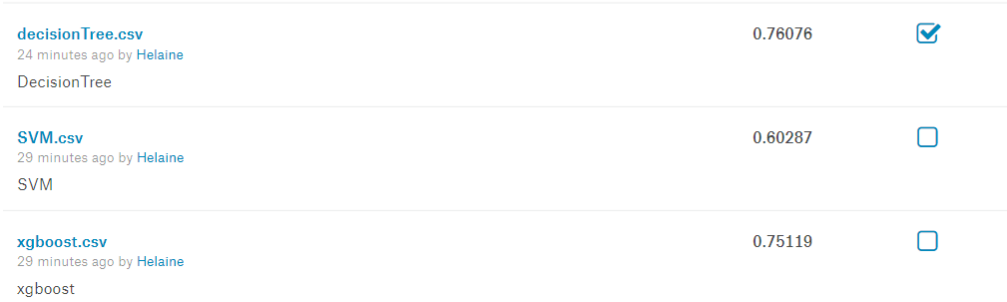


使用 SVM 分类器在测试集上的分类结果如图所示：



通过观察可视化结果我们可以发现，由于分类器都是根据数据特征向量与分类标签之间的关系进行建模，进行有监督的学习，三种分类器对数据的划分结果较为相似，且分布情况与训练集非常相似。且 DecisionTree 在此问题上对数据的划分能力优于 SVM 分类器，而 Xgboost 分类器同样作为一种基于树的方法，其对数据的区分结果与 DecisionTree 相似，但从图中可以看出其区分能力略优于 DecisionTree。

通过将对测试集的结果提交至 Kaggle 平台进行打分，我们获得三种分类器的分类性能如下:



从得分情况可以看出，三种分类器的分类结果为 DecisionTree > Xgboost > SVM，与我们观察分析的结果基本一致。而 Xgboost 虽然对数据的区分度优于 DecsionTree，也有可能因此而产生更多的误分类情况，因此在结果上稍逊于 DecisionTree。

3.4 聚类过程

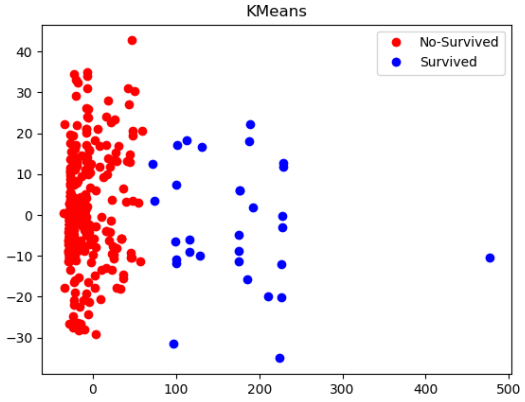
在经过预处理的数据集的基础上，分别采用 KMeans、Hierachical（层次聚类）、Spectral（谱聚类）三种方法进行聚类，由于本任务数据标签为二值标签，因此我们将数据聚为 2 类（n_clusters = 2），训练聚类模型的过程中不需要训练集的数据标签。最终我们使用训练好的模型对测试集进行预测。构建聚类模型的代码片段如下：

```python
def cluster_KMeans(train_X, test_X):
    clf = KMeans(n_clusters=2)
    clf.fit(train_X)
    predictions = clf.predict(test_X)
    return predictions

def cluster_Hierarchical(test_X):
    clf = AgglomerativeClustering(n_clusters=2)
    predictions = clf.fit_predict(test_X)
    return predictions

def cluster_Spectral(test_X):
    clf = SpectralClustering(n_clusters=2)
    predictions = clf.fit_predict(test_X)
    return predictions
```

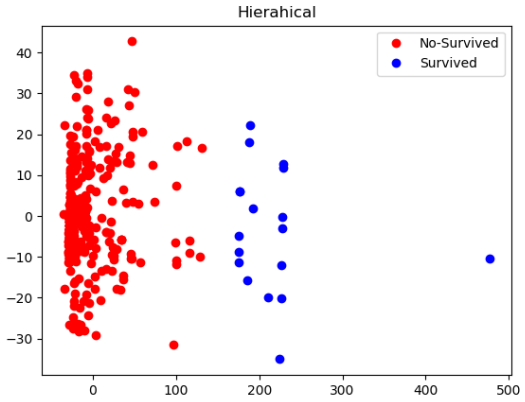
全部聚类模型的训练与预测位于“./src/clusters”。

3.5 聚类可视化结果

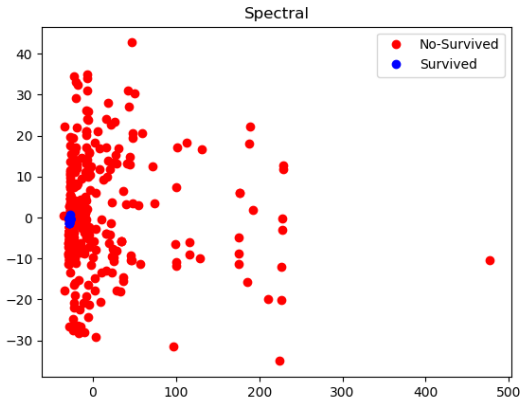
使用 KMeans 聚类模型在测试集上的聚类结果如图所示：



使用 Hierachical 聚类模型在测试集上的聚类结果如图所示：



使用 Spectral 聚类模型在测试集上的聚类结果如图所示:



通过观察可视化结果我们可以发现，KMeans 方法和 Hierachical 更接近于寻找到了两个簇中间的界限，对数据点的划分是比较明显的线性的划分；而 Spectral 方法由于是基于图的方法，因此其划分的数据点之间没有明显的区分界限。此外，由于 KMeans 方法是先在训练集上训练得到两个簇，再将测试集上的数据划分至两个簇中，因此其准确率可能会优于其他两个聚类方法。

通过将对测试集的结果提交至 Kaggle 平台进行打分，我们获得三种聚类模型的聚类结果如下:

Submission and Description	Public Score	Use for Final Score
Spectral.csv 13 minutes ago by Helaine Spectral	0.53110	☐
Hierachical.csv 22 minutes ago by Helaine Hierachical	0.65071	☐
KMeans.csv 23 minutes ago by Helaine Kmeans	0.66028	☐

我们可以看到结果为：KMeans > Hierachical > Spectral。且聚类模型的效果低于分类模型。因此我们验证了，在对于分类问题上，往往有监督的方法比无监督的方法能得到更好的效果。