

**院 系：计 算 机 学 院**

**实验课程： 数据挖掘**

**实验项目： 航空公司乘客满意调查分析**

**指导老师： 汤娜**

**开课时间：2020 ～ 2021 年度第 1学期**

**专 业：计算机科学与技术**

**班 级：3班**

**学 生：张升**

**学 号：20182131094**

**华南师范大学教务处**

# 编译环境

Spyder

# 数据来源

Kaggle

<https://www.kaggle.com/johndddddd/customer-satisfaction>

# 实验分析

该数据集在乘客满意度“satisfaction”这一列总共有两个取值：“satisfied”, “neutral or dissatisfaction”。所以对乘客满意度分类是属于二分类问题。

\*对比展示阶段，完善了神经网络的调参

# 实验过程

## 4.1数据预处理

**对分类数据进行编码**

|  |
| --- |
| #需要编码的列  features=['Satisfaction','Gender','Customer Type','Type of Travel','Class']  le=LabelEncoder()  for i in features:  df[i]=le.fit\_transform(df[i]) |

**处理缺失值**

|  |
| --- |
| cols=df.columns.tolist()  #使用众数代替缺失值  imp = SimpleImputer(missing\_values=np.nan, strategy='most\_frequent')  temp=imp.fit\_transform(df)  df=pd.DataFrame(temp,columns=cols) |

**数据探索**

预测几个对satisfaction影响较大的变量，分别把这几个变量和satisfaction的关系进行数据可视化。

|  |
| --- |
| #客户类型和满意程度  sns.barplot(x='Customer Type', y='Satisfaction',data=df)  plt.show()  # Customer Type:The customer type (0:Loyal customer, 1:disloyal customer) |
| #仓位和满意程度  sns.barplot(x='Class', y='Satisfaction',data=df)  plt.show()  # Class:Travel class in the plane of the passengers (0:Business, 1:Eco, 2:Eco Plus) |
| #满意程度在飞行距离中的分布  sns.boxplot(x='Satisfaction', y='Flight Distance',hue='Class',data=df)  plt.show() |
| #满意程度在出发延误时间中的分布  sns.boxplot(x='Satisfaction', y='Departure Delay in Minutes',hue='Type of Travel',data=df)  plt.ylim(0,200)  plt.show() |

**数据筛选降维**

1. 筛除线性相关的连续型变量

|  |
| --- |
| #用相关系数矩阵查看出发延误和到达延误的相关性  corrmatrix1 = df.iloc[:,-2:].corr(method='pearson')  #散点图证明  sns.pointplot(x='Departure Delay in Minutes',y='Arrival Delay in Minutes',data=df)    #由于两个变量线性相关，且到达延误在处理前存在缺失值，所以只取出发延误 |

1. 将剩下的连续型变量进行分箱。当所有变量都处理成分类变量后将他们进行卡方检验，筛除高相关性的变量
2. 将剩下的变量作为自变量，将satisfaction作为因变量，划分训练集和测试集

|  |
| --- |
| target = df['Satisfaction'] # 选取因变量  data=df.loc[:,['Gender','Customer Type','Food and drink',  'Inflight entertainment','Online support',  'On-board service','Leg room service',  'Checkin service','fl\_dis\_bins','dpt\_delay\_bins']] # 选取自变量  train\_data, test\_data, train\_target, test\_target = cross\_validation.train\_test\_split(data,target, test\_size=0.4, train\_size=0.6 ,random\_state=12345)  # 划分训练集和测试集 |

## 4.2决策树建模

(1) 使用网格搜索交叉验证的方式，决定决策树的最大深度和内部节点划分所需最小样本数

|  |
| --- |
| #定义超参数搜索网格，最大深度和内部节点划分所需最小样本数  param\_gridDT = {  'max\_depth':[10,11,12,13,14,15,16,17],  'min\_samples\_split':[8,10,12,14,16,18,20]  }  #进行网格搜索，采用交叉验证的方法，评价最优的超参数  DT = tree.DecisionTreeClassifier(criterion='entropy')  DTcv = GridSearchCV(estimator=DT, param\_grid=param\_gridDT,  scoring='roc\_auc', cv=4)  #最优模型保存在DTcv之中，进行模型训练  DTcv.fit(train\_data, train\_target)  #获取最优参数  DTbest\_params=DTcv.best\_params\_  #获取最佳模型评分  DTScore=DTcv.best\_score\_  **网格搜索最佳评分：0.9525276932762349** |

(2)用预处理阶段划分的测试集和训练集，绘制最优决策树模型的roc曲线，评估其拟合程度

|  |
| --- |
| # 查看模型预测结果  train\_est = DTcv.predict(train\_data) # 用模型预测训练集的结果  train\_est\_p=DTcv.predict\_proba(train\_data)[:,1] #用模型预测训练集的概率  test\_est=DTcv.predict(test\_data) # 用模型预测测试集的结果  test\_est\_p=DTcv.predict\_proba(test\_data)[:,1] # 用模型预测测试集的概率  #决策树对测试集分类准确度  DT\_accuracy=DTcv.score(test\_data, test\_target)  #绘制决策树roc曲线  fpr\_test, tpr\_test, th\_test = metrics.roc\_curve(test\_target, test\_est\_p)  fpr\_train, tpr\_train, th\_train = metrics.roc\_curve(train\_target, train\_est\_p)  plt.figure(figsize=[6,6])  plt.plot(fpr\_test, tpr\_test, color='blue')  plt.plot(fpr\_train, tpr\_train, color='red')  plt.title("DT ROC")  plt.show()  #红色训练集，蓝色测试集    **分类准确度：0.9544202722159922** |

## 4.3神经网络建模

(1) 使用网格搜索交叉验证的方式，决定神经网络的中间层层数和L2正则化参数

|  |
| --- |
| #定义搜索网格，隐藏层(层数，节点数)和L2正则化  param\_gridMLP={'hidden\_layer\_sizes':[(50,),(100,),(100,100)],'alpha':[0.0001,0.001,0.01,0.1]}  #进行网格搜索，采用交叉验证的方法，评价最优的超参数  MLP = MLPClassifier(activation='logistic')  MLPcv = GridSearchCV(estimator=MLP, param\_grid=param\_gridMLP,  scoring='roc\_auc', cv=4)  #神经网络最优模型保存在MLPcv中进行训练  MLPcv.fit(train\_data,train\_target)  #获取最优参数  MLPbest\_params=MLPcv.best\_params\_    #获取模型评分  MLPScore=MLPcv.best\_score\_  **网格搜索最佳评分：0.9575944175981318** |

(2) 用预处理阶段划分的测试集和训练集，绘制最优神经网络模型的roc曲线，评估其拟合程度

|  |
| --- |
| # 查看神经网络模型预测结果  train\_est1 = MLPcv.predict(train\_data) # 用模型预测训练集的结果  train\_est\_p1=MLPcv.predict\_proba(train\_data)[:,1] #用模型预测训练集的概率  test\_est1=MLPcv.predict(test\_data) # 用模型预测测试集的结果  test\_est\_p1=MLPcv.predict\_proba(test\_data)[:,1] # 用模型预测测试集的概率  #神经网络对测试集分类准确度  MLP\_accuracy=MLPcv.score(test\_data, test\_target)  #绘制神经网络roc曲线  fpr\_test1, tpr\_test1, th\_test1 = metrics.roc\_curve(test\_target, test\_est\_p1)  fpr\_train1, tpr\_train1, th\_train1 = metrics.roc\_curve(train\_target, train\_est\_p1)  plt.figure(figsize=[6,6])  plt.plot(fpr\_test1, tpr\_test1, color='blue')  plt.plot(fpr\_train1, tpr\_train1, color='red')  plt.title("MLP ROC")  plt.show()  #红色训练集，蓝色测试集    **分类准确度：0.9601086391466072** |

## 4.4模型比较

两个模型对测试集分类的roc曲线

|  |
| --- |
| plt.figure(figsize=[6,6])  plt.plot(fpr\_test1, tpr\_test1, color='red')  plt.plot(fpr\_test, tpr\_test, color='blue')  plt.title("model comparison")  plt.show()  #两种模型在测试集上的拟合程度，红色神经网络，蓝色决策树 |

分类准确度对比

决策树：0.9544202722159922

神经网络：0.9601086391466072